problem with integer data, it can be solved efficiently using the simplex method to compute a basic optimal solution, which the integrality theorem tells us will be integer valued.

6.1. König's Theorem. In addition to its importance in real-world optimization problems, the integrality theorem also has many applications to the branch of mathematics called combinatorics. We illustrate with just one example.

THEOREM 14.3. *König's Theorem. Suppose that there are n girls and n boys, that every girl knows exactly k boys, and that every boy knows exactly k girls. Then n marriages can be arranged with everybody knowing his or her spouse.*

Before proving this theorem it is important to clarify its statement by saying that the property of "knowing" is symmetric (for example, knowing in the biblical sense). That is, if a certain girl knows a certain boy, then this boy also knows this girl.

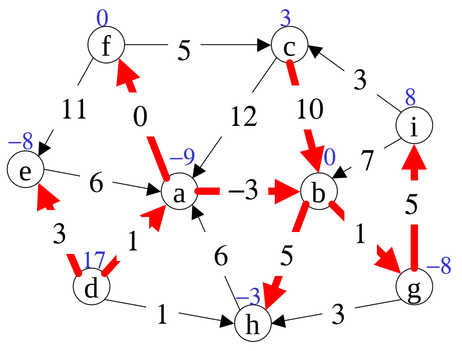
PROOF. Consider a network with nodes $g_1, g_2, \ldots, g_n, b_1, b_2, \ldots, b_n$ and an arc from g_i to b_j if girl i and boy j know each other. Assign one unit of supply to each girl node and a unit of demand to each boy node. Assign arbitrary objective coefficients to create a well-defined network flow problem. The problem is guaranteed to be feasible: just put a flow of $1/k$ on each arc (the polygamists in the group might prefer this nonintegral solution). By the integrality theorem, the problem has an integer-valued solution. Clearly, the flow on each arc must be either zero or one. Also, each girl node is the tail of exactly one arc having a flow of one. This arc points to her intended mate. $\square$

Exercises

In solving the following problems, the network pivot tool can be used to check your arithmetic:

www.princeton.edu/~rvdb/JAVA/network/nettool/netsimp.html

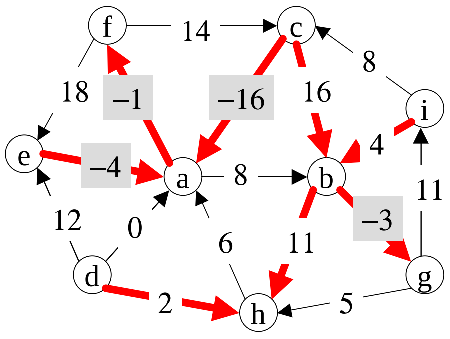
14.1 Consider the following network flow problem:



Numbers shown above the nodes are supplies (negative values represent demands) and numbers shown above the arcs are unit shipping costs. The darkened arcs form a spanning tree.

(a) Compute primal flows for each tree arc.
(b) Compute dual variables for each node.
(c) Compute dual slacks for each nontree arc.

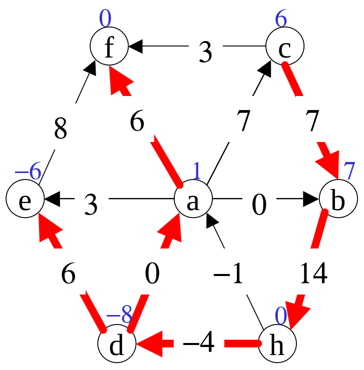
14.2 Consider the tree solution for the following minimum cost network flow problem:



The numbers on the tree arcs represent primal flows while numbers on the nontree arcs are dual slacks.

(a) Using the largest–coefficient rule in the dual network simplex method, what is the leaving arc?
(b) What is the entering arc?
(c) After *one* pivot, what is the new tree solution?

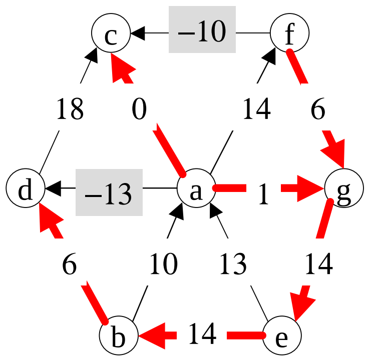
14.3 Consider the following network flow problem:



The numbers above the nodes are supplies (negative values represent demands) and numbers shown above the arcs are unit shipping costs. The darkened arcs form a spanning tree.

(a) Compute primal flows for each tree arc.
(b) Compute dual variables for each node.
(c) Compute dual slacks for each nontree arc.

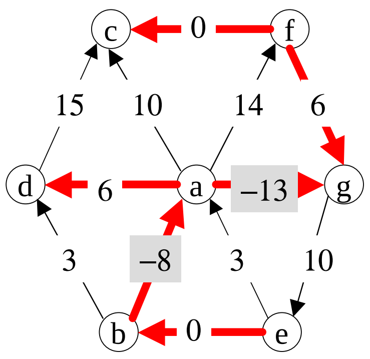
14.4 Consider the tree solution for the following minimum cost network flow problem:



The numbers on the tree arcs represent primal flows while numbers on the nontree arcs are dual slacks.

(a) Using the largest–coefficient rule in the primal network simplex method, what is the entering arc?
(b) What is the leaving arc?
(c) After *one* pivot, what is the new tree solution?

14.5 Consider the tree solution for the following minimum cost network flow problem:



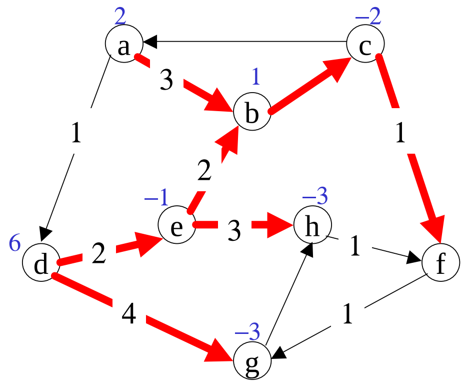
The numbers on the tree arcs represent primal flows while numbers on the nontree arcs are dual slacks.

(a) Using the largest–coefficient rule in the dual network simplex method, what is the leaving arc?

(b) What is the entering arc?

(c) After *one* pivot, what is the new tree solution?

14.6 Solve the following network flow problem starting with the spanning tree shown.



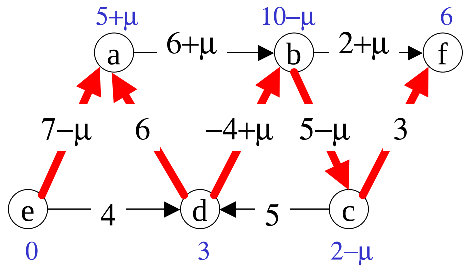
The numbers displayed next to nodes are supplies(+)/demands(−). Numbers on arcs are costs. Missing data should be assumed to be zero. The bold arcs represent an initial spanning tree.

14.7 Solve Exercise 2.11 using the self-dual network simplex method.

14.8 Using today's date (MMYY) for the seed value, solve 10 problems using the network simplex pivot tool:

www.princeton.edu/∼rvdb/JAVA/network/challenge/netsimp.html

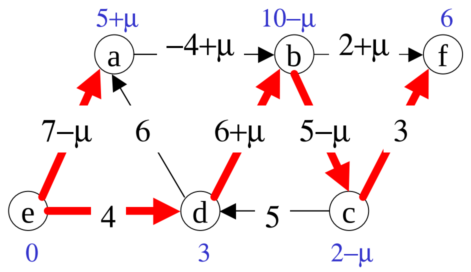
14.9 Consider the following tree solution for a minimum cost network flow problem:



As usual, bold arcs represent arcs on the spanning tree, numbers next to the bold arcs are primal flows, numbers next to non-bold arcs are dual slacks, and numbers next to nodes are dual variables.

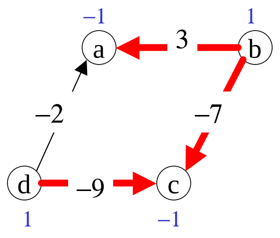
(a) For what values of μ is this tree solution optimal?
(b) What are the entering and leaving arcs?
(c) After *one* pivot, what is the new tree solution?
(d) For what values of μ is the new tree solution optimal?

14.10 Consider the following tree solution for a minimum cost network flow problem:



(a) For what values of μ is this tree solution optimal?
(b) What are the entering and leaving arcs?
(c) After *one* pivot, what is the new tree solution?
(d) For what values of μ is the new tree solution optimal?

14.11 Consider the following minimum cost network flow problem



As usual, the numbers on the arcs represent the flow costs and numbers at the nodes represent supplies (demands are shown as negative supplies). The arcs shown in bold represent a spanning tree. If the solution corresponding to this spanning tree is optimal prove it, otherwise find an optimal solution using this tree as the initial spanning tree.

14.12 Suppose that a square submatrix of $\tilde{A}$ is invertible. Show that the arcs corresponding to the columns of this submatrix form a spanning tree.

14.13 Show that a spanning tree on m nodes must have exactly $m - 1$ arcs.

14.14 Define an algorithm that takes as input a network and either finds a spanning tree or proves that the network is not connected.

14.15 Give an example of a minimum-cost network flow problem with all arc costs positive and the following counterintuitive property: if the supply at a particular source node and the demand at a particular sink node are simultaneously reduced by one unit, then the optimal cost increases.

14.16 Consider a possibly disconnected network $(\mathcal{N}, \mathcal{A})$. Two nodes i and j in $\mathcal{N}$ are said to be *connected* if there is a path from i to j (recall that paths can traverse arcs backwards or forwards). We write $i \sim j$ if i and j are connected.

(a) Show that "$\sim$" defines an *equivalence relation*. That is, it has the following three properties:

(i) (reflexivity) for all $i \in \mathcal{N}$, $i \sim i$;

(ii) (symmetry) for all $i, j \in \mathcal{N}$, $i \sim j$ implies that $j \sim i$;

(iii) (transitivity) for all $i, j, k \in \mathcal{N}$, $i \sim j$ and $j \sim k$ implies that $i \sim k$.

Using the equivalence relation, we can partition $\mathcal{N}$ into a collection of subsets of *equivalence classes* $\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_k$ such that two nodes are connected if and only if they belong to the same subset. The number k is called the number of *connected components*.

(b) Show that the rank of the node–arc incidence matrix A is exactly $m - k$ (recall that m is the number of rows of A).

14.17 One may assume without loss of generality that every node in a minimum cost network flow problem has at least two arcs associated with it. Why?

14.18 The sum of the dual slacks around any cycle is a constant. What is that constant?

14.19 *Planar Networks*. A network is called *planar* if the nodes and arcs can be laid out on the two-dimensional plane in such a manner that no two arcs cross each other (it is allowed to draw the arcs as curves if necessary). All of the networks encountered so far in this chapter have been planar. Associated with each planar network is a geometrically defined dual network. The purpose of this problem is to establish the following interesting fact:

A dual network simplex pivot is precisely a primal network simplex method applied to the dual network.

Viewed geometrically, the nodes of a planar graph are called *vertices* and the arcs are called *edges*. Consider a specific connected planar network. If one were to delete the vertices and the edges from the plane, one would be left with a disjoint collection of subsets of the plane. These subsets are called *faces*. Note that there is one unbounded face. It is a face just like

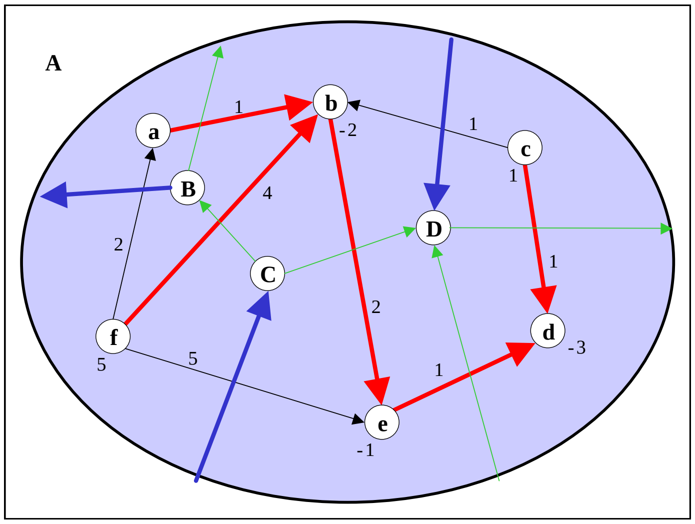


FIGURE 14.16. The primal network has nodes “a” through “f”. The corresponding dual network has nodes “A” through “D” (node “A” is “at infinity”). A primal spanning tree is shown. It consists of five arcs: (a,b), (f,b), (b,e), (e,d), and (c,d). The corresponding dual spanning tree consists of three arcs: (B,A), (A,C), and (D,A). Primal costs are shown along the primal arcs and supplies/demands are shown at the primal nodes.

the other bounded ones. An example of a connected planar network with its faces labeled A through D is shown in Figure 14.16.

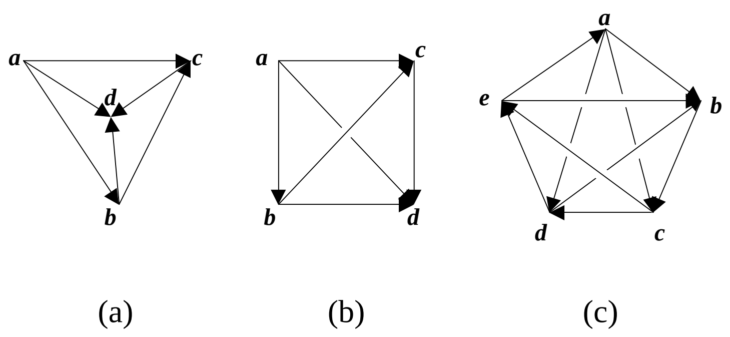
Dual nodes. Associated with each connected planar network is a *dual network* defined by interchanging vertices and faces. That is, place a dual vertex in the center of each primal face. Note: the dual vertex corresponding to the unbounded primal face could be placed anywhere in the unbounded face but we choose to put it *at infinity*. In this way, dual edges (defined next) that have a head or a tail at this node can run off to infinity in any direction.

Dual arcs. Connect with a dual edge any pair of dual nodes whose corresponding primal faces share an edge. Each dual edge crosses exactly one primal edge. The directionality of the dual edge is determined as follows: first, place a vector along the corresponding primal edge pointing in the direction of the primal arc, and then rotate it counterclockwise until it is tangent to the dual edge. The vector now defines the direction for the dual arc.

Dual spanning tree. Consider a spanning tree on the primal network and suppose that a primal–dual tree solution is given. We define a *spanning tree* on the dual network as follows. A dual edge is on the dual network's spanning tree if and only if the corresponding primal edge is not on the primal network's spanning tree.

Dual flows and dual dual-slacks. The numerical arc data for the dual network is inherited directly from the primal. That is, flows on the dual tree arcs are exactly equal to the dual slacks on the associated primal nontree arcs. And, the dual slacks on the the dual nontree arcs are exactly equal to the primal flows on the associated primal tree arcs. Having specified numerical data on the arcs of the dual network, it is fairly straightforward to determine values for supplies/demands at the nodes and shipping costs along the arcs that are consistent with these numerical values.

(a) Which of the following networks are planar:



(b) A network is called *complete* if there is an arc between every pair of nodes. If a complete network with m nodes is planar, then every network with m nodes is planar. Prove it.
(c) Show that a nonplanar network must have 5 or more nodes.
(d) As always, let m denote the number of nodes and let n denote the number of arcs in a network. Let f denote the number of faces in a planar network. Show by induction on f that $m = n - f + 2$.
(e) Show that the dual spanning tree defined above is in fact a spanning tree.
(f) Show that a dual pivot for a minimum cost network flow problem defined on the primal network is precisely the same as a primal pivot for the corresponding network flow problem on the dual network.
(g) Using the cost and supply/demand information given for the primal problem in Figure 14.16, write down the primal problem as a linear programming problem.

(h) Write down the dual linear programming problem that one derives algebraically from the primal linear programming problem.
(i) Using the spanning tree shown in Figure 14.16, compute the primal flows, dual variables, and dual slacks for the network flow problem associated with the primal network.
(j) Write down the flow and slacks for the network flow problem associated with the dual network.
(k) Find arc costs and node supplies/demands for the dual network that are consistent with the flows and slacks just computed.
(l) Write down the linear programming problem associated with the network flow problem on the dual network.

Notes

The classical reference is Ford & Fulkerson (1962). More recent works include the books by Christofides (1975), Lawler (1976), Bazaraa et al. (1977), Kennington & Helgason (1980), Jensen & Barnes (1980), Bertsekas (1991), and Ahuja et al. (1993).

The two "original" algorithms for solving minimum-cost network flow problems are the *network simplex method* developed by Dantzig (1951*a*) and the *primal–dual method* developed by Ford & Fulkerson (1958). The self-dual algorithm described in this chapter is neither of these. In fact, it resembles the "out-of-kilter" method described by Ford & Fulkerson (1962).