ternative approaches to the problem of capacity assignment and routing, see [Gav85], [NgH87], and [LeY89].

Section 5.5. The optimal routing problem considered is a special case of a multicommodity network flow problem. The problem arises also in the context of transportation networks in a form that is very similar to the one discussed here; see [AaM81], [Daf71], [Daf80], [DeK81], [FlN74], and [LaH84].

Section 5.6. The Frank–Wolfe method [FrW56] has been proposed in the context of the routing problem in [FGK73]. Problems 5.31 and 5.32 give a self-contained presentation of the convergence properties of this method as well as those of the gradient projection method.

Section 5.7. The gradient projection method was applied to multicommodity flow problems based on a path flow formulation in [Ber80]. Extensions were described in [BeG82], [BeG83], and [GaB84a]. A public domain FORTRAN implementation of the gradient projection method is given in [BGT84]. Distributed asynchronous versions of the algorithm are analyzed in [TsB86] and [Tsa89]. The optimal routing algorithm of [Gal77], which adjusts link flow fractions (see the end of Section 5.5), bears a relationship to the gradient projection method (see [BGG84]). Extensions of the algorithm are given in [Ber79a], [Gaf79], and [BGG84]. Computational and simulation results relating to this algorithm and its extensions may be found in [BGV79] and [ThC86].

Section 5.8. The comments of P. Humblet, one of the principal designers of the Codex network, were very helpful in preparing this section. Additional material can be found in [HuS86] and [HSS86].

PROBLEMS

5.1 Find a minimum weight spanning tree of the graph in Fig. 5.73 using the Prim–Dijkstra and the Kruskal algorithms.

5.2 Find the shortest path tree from every node to node 1 for the graph of Fig. 5.74 using the Bellman–Ford and Dijkstra algorithms.

5.3 The number shown next to each link of the network in Fig. 5.75 is the probability of the link failing during the lifetime of a virtual circuit from node A to node B. It is assumed that links fail independently of each other. Find the most reliable path from A to B, that

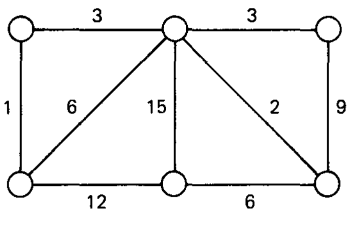

Figure 5.73 Graph for Problem 5.1.

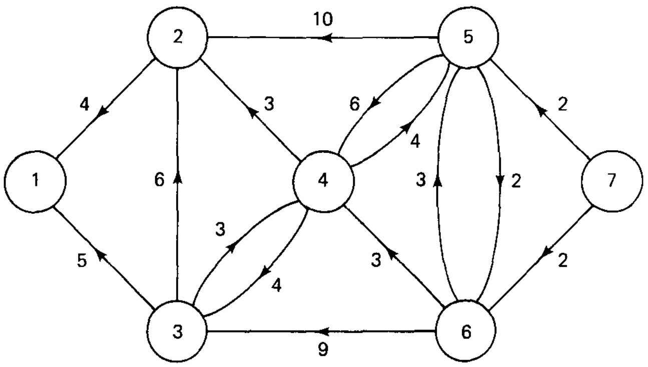

Figure 5.74 Graph for Problem 5.2.

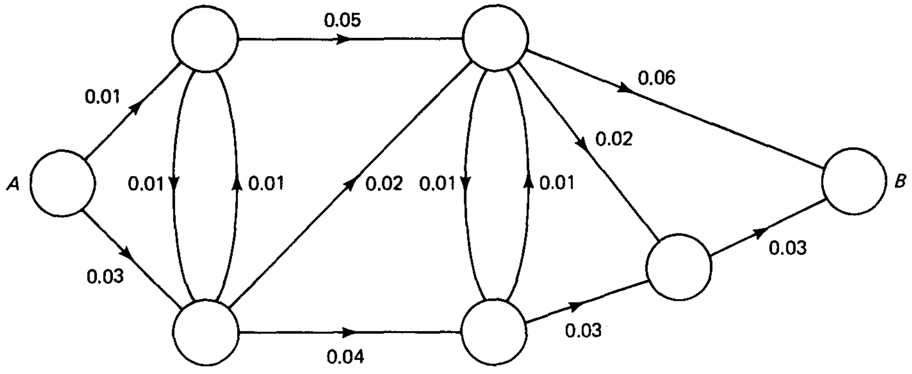

Figure 5.75 Graph for Problem 5.3.

is, the path for which the probability that all its links stay intact during the virtual circuit's lifetime is maximal. What is this probability?

5.4 A shortest path spanning tree (Section 5.2.3) differs from a minimum weight spanning tree in that the arc lengths refer to directed arcs, while in the minimum weight spanning tree problem, the arcs weights refer to undirected arcs or, equivalently, the arc weights are assumed equal in both directions. However, even if all arcs have equal length in both directions, a minimum weight spanning tree (for arc weights equal to the corresponding lengths) need not be a shortest path spanning tree. To see this, consider a network with three nodes A, B, and C, and three bidirectional arcs AB, BC, and CA. Choose a weight for each arc so that the minimum weight spanning tree is not a shortest path tree with root node C. (The length of an arc in each of its two directions is equal to the arc weight.)

5.5 Consider Example 5.4.

(a) Repeat the example with $N = 6$ instead of $N = 16$. (Nodes $1, 2, 4$, and 5 send 1 unit to node 6, while node 3 sends ϵ with $0 < \epsilon \ll 1$.)

(b) Repeat part (a) with the difference that the length d_{ij} of link (i, j) is $\alpha + F_{ij}$ for $\alpha = 1$ (instead of F_{ij}). Consider all possible choices of initial routing.

(c) What is the minimum value of α for which the shortest paths of all nodes except node 3 eventually stay constant regardless of the choice of initial routing?

(d) Repeat part (a) with the difference that at each iteration after the first, the length of each link is the average of the link arrival rates corresponding to the current and the preceding routings (instead of the arrival rates at just the current routing).

5.6 Consider the shortest path problem assuming that all cycles not containing node 1 have nonnegative length. Let $\tilde{D}_j$ be the shortest path distances corresponding to a set of link lengths $\tilde{d}_{ij}$, and let d_{ij} be a new set of link lengths. For $k = 1, 2, \ldots$, define

$$N_1 = \left\{ i \neq 1 \mid \text{for some arc } (i,j) \text{ with } d_{ij} > \tilde{d}_{ij} \text{ we have } \tilde{D}_i = \tilde{d}_{ij} + \tilde{D}_j \right\}$$

$$N_{k+1} = \left\{ i \neq 1 \mid \text{for some arc } (i,j) \text{ with } j \in N_k \text{ we have } \tilde{D}_i = \tilde{d}_{ij} + \tilde{D}_j \right\}$$

(a) Show that the Bellman–Ford algorithm starting from the initial conditions $D_1^0 = 0$, $D_i^0 = \tilde{D}_i$ for $i \notin \cup_k N_k \cup \{1\}$, and $D_i^0 = \infty$ for $i \in \cup_k N_k$ terminates after at most N iterations.

(b) Based on part (a), suggest a heuristic method for alleviating the "bad news phenomenon" of Fig. 5.36, which arises in the asynchronous Bellman–Ford algorithm when some link lengths increase. *Hint*: Whenever the length of a link on the current shortest path increases, the head node of the link should propagate an estimated distance of ∞ along the shortest path tree in the direction away from the destination.

5.7 Consider the Bellman–Ford algorithm assuming that all cycles not containing node 1 have nonnegative length. For each node $i \neq 1$, let (i, j_i) be an arc such that j_i attains the minimum in the equation

$$D_i^{h_i} = \min_j \left[d_{ij} + D_j^{h_i - 1} \right]$$

where h_i is the largest h such that $D_i^h \neq D_i^{h-1}$. Consider the subgraph consisting of the arcs (i, j_i), for $i \neq 1$, and show that it is a spanning tree consisting of shortest paths. *Hint*: Show that $h_i > h_{j_i}$.

5.8 Consider the shortest path problem. Show that if there is a cycle of zero length not containing node 1, but no cycles of negative length not containing node 1, Bellman's equation has more than one solution.

5.9 Suppose that we have a directed graph with no directed cycles. We are given a length d_{ij} for each directed arc (i, j) and we want to compute a shortest path to node 1 from all other nodes, assuming there exists at least one such path. Show that nodes $2, 3, \ldots, N$ can be renumbered so that there is an arc from i to j only if $i > j$. Show that once the nodes are renumbered, Bellman's equation can be solved with $O(N^2)$ operations at worst.

5.10 Consider the shortest path problem from every node to node 1 and Dijkstra's algorithm.

(a) Assume that a positive lower bound is known for all arc lengths. Show how to modify the algorithm to allow more than one node to enter the set of permanently labeled nodes at each iteration.

(b) Suppose that we have already calculated the shortest path from every node to node 1, and assume that a single arc length *increases*. Modify Dijkstra's algorithm to recalculate as efficiently as you can the shortest paths.

5.11 *A Generic Single-Destination Multiple-Origin Shortest Path Algorithm.* Consider the shortest path problem from all nodes to node 1. The following algorithm maintains a list of nodes V and a vector $D = (D_1, D_2, \ldots, D_N)$, where each D_j is either a real number or ∞.

Initially,

$$V = \{1\}$$

$$D_1 = 0, \qquad D_i = \infty, \qquad \text{for all } i \neq 1$$

The algorithm proceeds in iterations and terminates when V is empty. The typical iteration (assuming that V is nonempty) is as follows:

Remove a node j from V. For each incoming arc $(i, j) \in \mathcal{A}$, with $i \neq 1$, if $D_i > d_{ij} + D_j$, set

$$D_i = d_{ij} + D_j$$

and add i to V if it does not already belong to V.

(a) Show that at the end of each iteration: (1) if $D_j < \infty$, then D_j is the length of some path starting at j and ending at 1, and (2) if $j \notin V$, then either $D_j = \infty$ or else

$$D_i \leq d_{ij} + D_j, \qquad \text{for all } i \text{ such that } (i, j) \in \mathcal{A}$$

(b) If the algorithm terminates, then upon termination, for all $i \neq 1$ such that $D_i < \infty$, D_i is the shortest distance from i to 1 and

$$D_i = \min_{(i,j)\in\mathcal{A}} [d_{ij} + D_j]$$

Furthermore, $D_i = \infty$ for all i such that there is no path from i to 1.

(c) If the algorithm does not terminate, there exist paths of arbitrarily small length from at least one node i to node 1.

(d) Show that if at each iteration the node j removed from V satisfies $D_j = \min_{i\in V} D_i$, the algorithm is equivalent to Dijkstra's algorithm.

5.12 *A Generic Single-Destination Single-Origin Shortest Path Algorithm.* Consider the problem of finding a shortest path from a given node t to node 1. Suppose that for all i we have an underestimate u_i of the shortest distance from t to i. The following algorithm maintains a list of nodes V and a vector $D = (D_1, D_2, \ldots, D_N)$, where each D_i is either a real number or ∞. Initially,

$$V = \{1\}$$

$$D_1 = 0, \qquad D_i = \infty, \qquad \text{for all } i \neq 1$$

The algorithm proceeds in iterations and terminates when V is empty. The typical iteration (assuming that V is nonempty) is as follows:

Remove a node j from V. For each incoming arc $(i, j) \in \mathcal{A}$, with $i \neq 1$, if $\min\{D_i, D_t - u_i\} > d_{ij} + D_j$, set

$$D_i = d_{ij} + D_j$$

and add i to V if it does not already belong to V.

(a) If the algorithm terminates, then upon termination, either $D_t < \infty$, in which case D_t is the shortest distance from t to 1, or else there is no path from t to 1.

(b) If the algorithm does not terminate, there exist paths of arbitrarily small length from at least one node i to node 1.

5.13 Consider the second flooding algorithm of Section 5.3.2. Suppose that there is a known upper bound on the time an update message requires to reach every node connected with the originating node. Devise a scheme based on an age field to supplement the algorithm so that it works correctly even if the sequence numbers can wrap around due to a memory or communication error. *Note*: Packets carrying an age field should be used only in exceptional circumstances after the originating node detects an error.

5.14 Modify the SPTA so that it can be used to broadcast one-directional link information other than status throughout the network (cf. the remarks at the end of Section 5.3.3). The link information is collected by the start node of the link. Justify the modified algorithm. *Hint*: Add to the existing algorithm additional main and port tables to hold the directional link information. Formulate update rules for these tables using the node labels provided by the main topology update algorithm of Section 5.3.3.

5.15 (a) Give an example where the following algorithm for broadcasting topological update information fails. Initially, all nodes know the correct status of all links.

Update rules:

1. When an adjacent link changes status, the node sends the new status in a message on all operating adjacent links.
2. When a node receives a message about a nonadjacent link which differs from its view of the status of the link, it changes the status of the link in its topology table. It also sends the new status in a message on all operating adjacent links *except* the one on which it received the message.
3. When a node receives a message about an adjacent link which differs from its view of the status of the link, it sends the correct status on the link on which it received the message.

Hint: Failure occurs with simple network topologies, such as the one shown in Fig. 5.43.

(b) Give an example of failure when the word "except" in rule 2 is changed to "including."

5.16 Describe what, if anything, can go wrong in the topology broadcast algorithms of Section 5.3 if the assumption that messages are received across links in the order transmitted is relaxed. Consider the ARPANET flooding algorithm, the two algorithms of Section 5.3.2, and the SPTA of Section 5.3.3.

5.17 *Distributed Computation of the Number of Nodes in a Network.* Consider a strongly connected communication network with N nodes and A (bidirectional) links. Each node knows its identity and the set of its immediate neighbors but not the network topology. Node 1 wishes to determine the number of nodes in the network. As a first step, it initiates an algorithm for finding a *directed, rooted spanning tree with node 1 as the root.* By this we mean a tree each link (i, k) of which is directed and oriented toward node 1 along the unique path on the tree leading from i to 1 (see the example tree shown in Fig. 5.76).

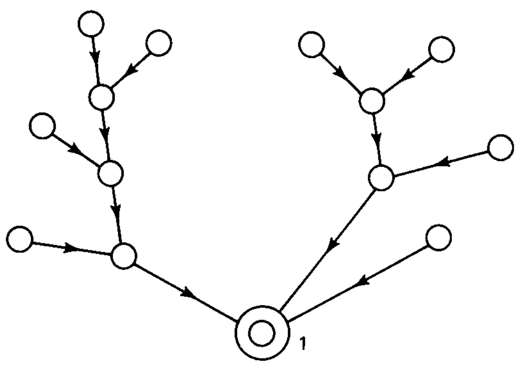

Figure 5.76 Example tree for Problem 5.17.

(a) Devise a distributed algorithm involving exchange of messages between nodes that constructs such a tree. The algorithm is initiated by node 1 and should involve no more than $O(A)$ message transmissions. Communication along any link is assumed error-free. At the end of the algorithm, the end nodes of each link should know whether the link is part of the tree, and, if so, they should know its direction.

(b) Supplement the algorithm derived in part (a) by another algorithm involving no more than $O(N)$ message transmissions by means of which node 1 gets to know N.

(c) Assuming that each message transmission takes an equal amount of time T, derive an upper bound for the time needed to complete the algorithms in parts (a) and (b).

5.18 Consider the minimum weight spanning tree problem. Show that the following procedure implements the Prim–Dijkstra algorithm and that it requires $O(N^2)$ arithmetic operations. Let node 1 be the starting node, and initially set $P = \{1\}$, $T = \emptyset$, $D_1 = 0$, and $D_j = w_{1j}$, $a_j = 1$ for $j \neq 1$.

Step 1. Find $i \notin P$ such that

$$D_i = \min_{j \notin P} D_j$$

and set $T := T \cup \{(j, a_i)\}$, $P := P \cup \{i\}$. If P contains all nodes, stop.

Step 2. For all $j \notin P$, if $w_{ij} < D_j$, set $D_j := w_{ij}$, $a_j := i$. Go to step 1.

Note: Observe the similarity with Dijkstra's algorithm for shortest paths.

5.19 Use Kleitman's algorithm to prove or disprove that the network shown in Fig. 5.77 is 3-connected. What is the maximum k for which the network is k-connected?

5.20 Suppose you had an algorithm that would find the maximum k for which two nodes in a graph are k-connected (*e.g.*, the max-flow algorithm test illustrated in Fig. 5.57). How would you modify Kleitman's algorithm to find the maximum k for which the graph is k-connected? Apply your modified algorithm to the graph of Fig. 5.77.

5.21 Use the Essau–Williams heuristic algorithm to find a constrained MST for the network shown in Fig. 5.78. Node 0 is the central node. The link weights are shown next to the links. The input flows from each node to the concentrator are shown next to the arrows. All link capacities are 10.

5.22 *MST Algorithm [MaP88].* We are given an undirected graph $G = (N, A)$ with a set of nodes $N = \{1, 2, \ldots, n\}$, and a weight a_{ij} for each arc (i, j) $(a_{ij} = a_{ji})$. We refer to the maximum over all the weights of the arcs contained in a given walk as the *critical weight* of the walk.

(a) Show that an arc (i, j) belongs to a minimum weight spanning tree (MST) if and only if the critical weight of every walk starting at i and ending at j is greater or equal to a_{ij}.

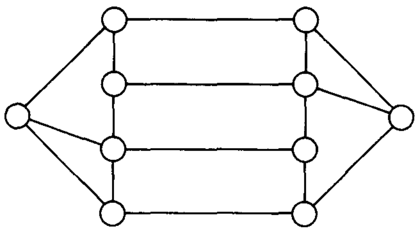

Figure 5.77 Network for Problem 5.19.

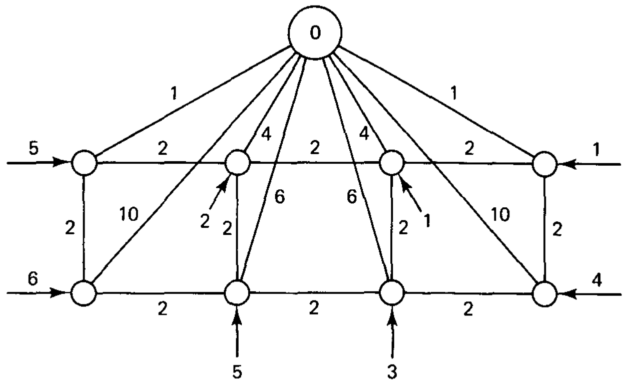

Figure 5.78 Network for Problem 5.21.

(b) Consider the following iterative algorithm:

$$x_{ij}^0 = \begin{cases} a_{ij}, & \text{if } (i,j) \in A, \\ \infty, & \text{otherwise,} \end{cases}$$

$$x_{ij}^{k+1} = \begin{cases} \min\left\{x_{ij}^k,\ \max\left\{x_{i(k+1)}^k, x_{(k+1)j}^k\right\}\right\}, & \text{if } j \neq i, \\ \infty, & \text{otherwise.} \end{cases}$$

Show that x_{ij}^k is the minimum over all critical weights of walks that start at i, end at j, and use only nodes 1 to k as intermediate nodes.

5.23 *Suboptimal Selective Broadcasting.* We are given a connected undirected graph $G = (N, A)$ with a nonnegative weight a_{ij} for each arc $(i,j) \in A$. Consider a tree in G which spans a given subset of nodes $\tilde{N}$ and has minimum weight over all such trees. Let W^* be the weight of this tree. Consider the graph $I(G)$, which has node set $\tilde{N}$ and is complete (has an arc connecting every pair of its nodes). Let the weight for each arc (i,j) of $I(G)$ be equal to the shortest distance in the graph G from the node $i \in \tilde{N}$ to the node $j \in \tilde{N}$. [Here, G is viewed as a directed graph with bidirectional arcs and length a_{ij} for each arc $(i,j) \in A$ in each direction.] Let T be a minimum weight spanning tree of $I(G)$.

(a) Show that the weight of T is no greater than $2W^*$. *Hint:* A *tour* in a graph is a cycle with no repeated nodes that passes through all the nodes. Consider a minimum weight tour in $I(G)$. Show that the weight of this tour is no less than the weight of T and no more than $2W^*$.

(b) Provide an example where the weight of T lies strictly between W^* and $2W^*$.

(c) Develop an algorithm for selective broadcasting from one node of $\tilde{N}$ to all other nodes of $\tilde{N}$ using T. Show that this algorithm comes within a factor of 2 of being optimal based on the given weights of the arcs of G.

5.24 Show that the necessary optimality condition of Section 5.5,

$$x_p^* > 0 \quad \Rightarrow \quad \frac{\partial D(x^*)}{\partial x_{p'}} \geq \frac{\partial D(x^*)}{\partial x_p}, \qquad \text{for all } p' \in P_w$$

implies that for all feasible path flow vectors $x = \{x_p\}$, we have

$$\sum_{p \in P_w} \left(x_p - x_p^*\right) \frac{\partial D(x^*)}{\partial x_p} \geq 0, \qquad \text{for all OD pairs } w$$

[The latter condition is sufficient for optimality of x^* when $D(x)$ is a convex function, that is,

$$D\big(\alpha x + (1-\alpha)x'\big) \le \alpha D(x) + (1-\alpha)D(x')$$

for all feasible x and x' and all α with $0 \le \alpha \le 1$. To see this, note that if $D(x)$ is convex, we have

$$D(x) \ge D(x^*) + \sum_{w\in W}\sum_{p\in P_w}\left(x_p - x_p^*\right)\frac{\partial D(x^*)}{\partial x_p}$$

for all x ([Lue84], p. 178), so when the shortest path condition above holds, x^* is guaranteed to be optimal.] *Hint*: Let $D_w^* = \min_{p\in P_w} \partial D(x^*)/\partial x_p$. We have for every feasible x,

$$0 = \sum_{p\in P_w}(x_p - x_p^*)D_w^* \le \sum_{\{p\in P_w | x_p > x_p^*\}}(x_p - x_p^*)\frac{\partial D(x^*)}{\partial x_p} + \sum_{\{p\in P_w | x_p < x_p^*\}}(x_p - x_p^*)D_w^*$$

5.25 Consider the network shown in Fig. 5.79 involving one OD pair with a given input rate r and three links with capacities C_1, C_2, and C_3, respectively. Assume that $C_1 = C_3 = C$, and $C_2 > C$, and solve the optimal routing problem with cost function based on the $M/M/1$ delay approximation and r between 0 and $2C + C_2$.

5.26 Consider the problem of finding the optimal routing of one unit of input flow in a single-OD-pair, three-link network for the cost function

$$D(x) = \frac{1}{2}\left[(x_1^2) + 2(x_2^2) + (x_3^2)\right] + 0.7x_3$$

Mathematically, the problem is

$$\begin{aligned}&\text{minimize } D(x)\\ &\text{subject to } x_1 + x_2 + x_3 = 1\\ &\qquad\qquad x_1, x_2, x_3 \ge 0\end{aligned}$$

(a) Show that $x_1^* = 2/3$, $x_2^* = 1/3$, and $x_3^* = 0$ is the optimal solution.

(b) Carry out several iterations of the Frank–Wolfe method and the projection method starting from the point (1/3, 1/3, 1/3). Do enough iterations to demonstrate a clear trend in rate of convergence. (Use a computer for this if you wish.) Plot the successive iterates on the simplex of feasible flows.

5.27 *Extensions of the Gradient Projection Method*

(a) Show how the optimality condition of Section 5.5 and the gradient projection method can be extended to handle the situation where the cost function is a general twice differentiable function of the path flow vector x.

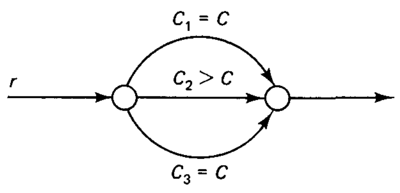

Figure 5.79 Network for Problem 5.25.

(b) Discuss the special case where the cost of link (i, j) is $D_{ij}(\tilde{F}_{ij}, F_{ij})$ where

$$\tilde{F}_{ij} = \sum_{k=1}^{M} p_k F_{ij}^k, \qquad F_{ij} = \sum_{k=1}^{M} F_{ij}^k$$

p_k is a positive scalar weighting factor for priority class k, and F_{ij}^k is the total flow of priority class k crossing link (i, j) (cf. the Codex algorithm cost function).

5.28 *Optimal Broadcast Routing along Spanning Trees.* Consider the optimal routing problem of Section 5.4. Suppose that in addition to the regular OD pair input rates r_w that are to be routed along directed paths, there is additional traffic R to be broadcast from a single node, say 1, to all other nodes along one or more directed spanning trees rooted at node 1. The spanning trees to be used for routing are to be selected from a given collection T, and the portion of R to be broadcast along each is subject to optimization.

(a) Characterize the path flow rates and the portions of R broadcast along spanning trees in T that minimize a total link cost, such as the one given by Eq. (5.29) in Section 5.4.

(b) Extend the gradient projection method to solve this problem.

(c) Extend your analysis to the case where there are several root nodes and there is traffic to be broadcast from each of these nodes to all other nodes.

Hint: Consider the first derivative total weights of spanning trees in addition to the first derivative lengths of paths.

5.29 *Optimal Routing with VCs Using the Same Links in Both Directions.* Consider the optimal routing problem of Section 5.4 with the additional restriction that the traffic originating at node i and destined for node i' must use the same path (in the reverse direction) as the traffic originating at i' and destined for i. In particular, assume that if w and w' are the OD pairs (i, i') and (i', i), respectively, there is a proportionality constant $c_{ww'}$ such that if $p \in P_w$ is a path and $p' \in P_{w'}$ is the reverse path, and x_p and $x_{p'}$ are the corresponding path flows, then $x_p = c_{ww'} x_{p'}$. Derive the conditions characterizing an optimal routing, and appropriately extend the gradient projection method.

5.30 *Use of the $M/M/1$-Based Cost Function with the Gradient Projection Method.* When the cost function

$$\sum_{(i,j)} D_{ij}(F_{ij}) = \sum_{(i,j)} \frac{F_{ij}}{C_{ij} - F_{ij}}$$

is minimized using the gradient projection method, there is potential difficulty due to the fact that some link flow F_{ij} may exceed the capacity C_{ij}. One way to get around this is to replace D_{ij} with a function $\tilde{D}_{ij}$ that is identical with D_{ij} for F_{ij} in the interval $[0, \rho C_{ij}]$, where ρ is some number less than one (say, $\rho = 0.99$), and is quadratic for $F_{ij} > \rho C_{ij}$. The quadratic function is chosen so that D_{ij} and $\tilde{D}_{ij}$ have equal values and first two derivatives at the point $F_{ij} = \rho C_{ij}$. Derive the formula for $\tilde{D}_{ij}$. Show that if the link flows F_{ij}^* that minimize

$$\sum_{(i,j)} D_{ij}(F_{ij})$$

result in link utilizations not exceeding ρ [*i.e.*, $F_{ij}^* \leq \rho C_{ij}$ for all (i, j)], then F_{ij}^* also minimize

$$\sum_{(i,j)} \tilde{D}_{ij}(F_{ij})$$

5.31 *Convergence of the Frank–Wolfe Algorithm.* The purpose of this problem and the next one is to guide the advanced reader through convergence proofs of the Frank–Wolfe and the gradient projection methods. Consider the problem

$$\text{minimize } f(x)$$

$$\text{subject to } x \in X$$

where f is a differentiable function of the n-dimensional vector x and X is a closed, bounded, convex set. Assume that the gradient of f satisfies

$$|\nabla f(x) - \nabla f(y)| \le L|x - y|, \quad \text{for all } x, y \in X$$

where L is a positive constant and $|z|$ denotes the Euclidean norm of a vector z, that is,

$$|z| = \sqrt{\textstyle\sum_{i=1}^{n} (z_i)^2}$$

(This assumption can be shown to hold if $\nabla^2 f$ exists and is continuous over X.)

(a) Show that for all vectors x, Δx, and scalars α, such that $x \in X$, $x + \Delta x \in X$, $\alpha \in [0, 1]$,

$$f(x + \alpha \Delta x) \le f(x) + \alpha \nabla f(x)^T \Delta x + \frac{\alpha^2 L}{2} |\Delta x|^2$$

where the superscript T denotes transpose. *Hint*: Use the Taylor formula

$$f(x + y) = f(x) + \nabla f(x)^T y + \int_0^1 \left[\nabla f(x + ty) - \nabla f(x)\right]^T y \, dt$$

(b) Use part (a) to show that if Δx is a descent direction at x [*i.e.*, $\nabla f(x)^T \Delta x < 0$], then

$$\min_{\alpha \in [0,1]} f(x + \alpha \Delta x) \le f(x) + \delta$$

where

$$\delta = \begin{cases} \frac{1}{2} \nabla f(x)^T \Delta x, & \text{if} \quad \nabla f(x)^T \Delta x + L|\Delta x|^2 < 0 \\ -\frac{|\nabla f(x)^T \Delta x|^2}{2LR^2}, & \text{otherwise} \end{cases}$$

where R is the diameter of X, that is,

$$R = \max_{x, y \in X} |x - y|$$

Hint: Minimize over $\alpha \in [0, 1]$ in both sides of the inequality of part (a).

(c) Consider the Frank–Wolfe method

$$x^{k+1} = x^k + \alpha^k \Delta x^k$$

where $x^0 \in X$ is the starting vector, Δx^k solves the problem

$$\text{minimize } \nabla f(x^k)^T \Delta x$$

$$\text{subject to } x^k + \Delta x \in X$$

and α^k is a minimizing stepsize

$$f(x^k + \alpha^k \Delta x^k) = \min_{\alpha\in[0,1]} f(x^k + \alpha \Delta x^k)$$

Show that every limit x^* of the sequence $\{x^k\}$ satisfies the optimality condition

$$\nabla f(x^*)^T (x - x^*) \geq 0, \qquad \text{for all } x \in X$$

Hint: Argue that if $\{x^k\}$ has a limit point and δ^k corresponds to x^k as in part (b), then $\delta^k \to 0$, and therefore also $\nabla f(x^k)^T \Delta x^k \to 0$. Taking the limit in the relation $\nabla f(x^k)^T \Delta x \leq \nabla f(x^k)^T (x - x^k)$ for all $x \in X$, argue that a limit point x^* must satisfy $0 \leq \nabla f(x^*)^T (x - x^*)$ for all $x \in X$.

5.32 *Convergence of the Gradient Projection Method.* Consider the minimization problem and the assumptions of Problem 5.31. Consider also the iteration

$$x^{k+1} = x^k + \alpha^k (\bar{x}^k - x^k)$$

where $\bar{x}^k$ is the projection of $x^k - s\nabla f(x^k)$ on X and s is some fixed positive scalar, that is, $\bar{x}^k$ solves

$$\text{minimize } \left|x - x^k + s\nabla f(x^k)\right|^2$$

$$\text{subject to } x \in X$$

(a) Assume that a^k is a minimizing stepsize

$$f\left[x^k + \alpha^k(\bar{x}^k - x^k)\right] = \min_{\alpha\in[0,1]} f\left[x^k + \alpha(\bar{x}^k - x^k)\right]$$

and show that every limit point x^* of $\{x^k\}$ satisfies the optimality condition

$$\nabla f(x^*)^T (x - x^*) \geq 0, \qquad \text{for all } x \in X$$

Hint: Follow the hints of Problem 5.31. Use the following necessary condition satisfied by the projection

$$\left[x^k - s\nabla f(x^k) - \bar{x}^k\right]^T (x - \bar{x}^k) \leq 0, \qquad \text{for all } x \in X$$

to show that $s\nabla f(x^k)^T(\bar{x}^k - x^k) \leq -|\bar{x}^k - x^k|^2$.

(b) Assume that $a^k = 1$ for all k. Show that if $s < 2/L$, the conclusion of part (a) holds.

5.33 *Gradient Projection Method with a Diagonal Scaling Matrix.* Consider the problem

$$\text{minimize } f(x)$$

$$\text{subject to } x \geq 0$$

of Section 5.7. Show that the iteration

$$x_i^{k+1} = \max\left\{0, x_i^k - \alpha^k b_i \frac{\partial f(x^k)}{\partial x_i}\right\}, \qquad i = 1, \ldots, n$$

with b_i some positive scalars, is equivalent to the gradient projection iteration

$$y_i^{k+1} = \max\left\{0, y_i^k - \alpha^k \frac{\partial h(y^k)}{\partial y_i}\right\}, \qquad i = 1, \ldots, n$$

where the variables x_i and y_i are related by $x_i = \sqrt{b_i}y_i$, and $h(y) = f(Ty)$, and T is the diagonal matrix with the i^{th} diagonal element equal to $\sqrt{b_i}$.

5.34 This problem illustrates the relation between optimal routing and adaptive shortest path routing for virtual citcuits; see also Section 5.2.5 and [GaB83]. Consider a virtual circuit (VC) routing problem involving a single origin–destination pair and two links as shown in the figure. Assume that a VC arrives at each of the times $0, \tau, 2\tau, \ldots,$ and departs exactly H time units later. Each VC is assigned to one of the two links and remains assigned on that link for its entire duration. Let $N_i(t)$, $i = 1, 2$, be the number of VCs on link i at time t. We assume that initially the system is empty, and that an arrival and/or departure at time t is not counted in $N_i(t)$, so, for example, $N_i(0) = 0$. The routing algorithm is of the adaptive shortest path type and works as follows: For $kT \le t < (k+1)T$, a VC that arrives at time t is routed on link 1 if $\gamma_1 N_1(kT) \le \gamma_2 N_2(kT)$ and is routed on link 2 otherwise. Here T, γ_1, and γ_2 are given positive scalars.

Assume that $T \le H$ and that τ divides evenly H. Show that:

(a) $N_1(t) + N_2(t) = H/\tau$ for all $t > H$.

(b) For all $t > H$ we have

$$\frac{|N_1(t) - N_1^*|}{N_1^*} \le \frac{\gamma_1 + \gamma_2}{\gamma_2}\frac{T}{H}, \qquad \frac{|N_2(t) - N_2^*|}{N_2^*} \le \frac{\gamma_1 + \gamma_2}{\gamma_1}\frac{T}{H}$$

where (N_1^*, N_2^*) solve the optimal routing problem

$$\text{minimize } \gamma_1 N_1^2 + \gamma_2 N_2^2$$

$$\text{subject to } N_1 + N_2 = H/\tau, \quad N_1 \ge 0,\ N_2 \ge 0$$

Hint: Show that for $t > H$ we have

$$\gamma_1 N_1(t) \le \gamma_2 N_2(t) \qquad \Longleftrightarrow \qquad N_1(t) \le N_1^*$$

5.35 *Virtual Circuit Routing by Flooding.* In some networks where it may be hard to keep track of the topology as it changes (*e.g.*, packet radio networks), it is possible to use flooding to set up virtual circuits. The main idea is that a node m, which wants to set up a virtual circuit to node n, should flood an "exploratory" packet through the network. A node that rebroadcasts this packet stamps its ID number on it so that when the destination node n receives an exploratory packet, it knows the route along which it came. The destination node can then choose one of the potentially many routes carried by the copies of the exploratory packet received and proceed to set up the virtual circuit. Describe one or more flooding protocols that will make this process workable. Address the issues of indefinite or excessive message circulation in the network, appropriate numbering of exploratory packets, unpredictable link failures and delays, potential confusion between the two end nodes of virtual circuits, and so on.

5.36 Consider the optimal routing problem of Section 5.5. Assume that each link cost is chosen to be the same function $D(F)$ of the link flow F, where the first link cost derivative at zero flow $D'(0)$ is positive.

(a) For sufficiently small values of origin–destination pair input rates r_w show that optimal routings use only minimum-hop paths from origins to destinations.

(b) Construct an example showing that optimal routings do not necessarily have the minimum-hop property for larger values of r_w.

5.37 Consider the Frank–Wolfe method with the stepsize of Eq. (5.63) in Section 5.6. Describe an implementation as a distributed synchronous algorithm involving communication from origins to links and the reverse. At each iteration of this algorithm, each origin should calculate the stepsize of Eq. (5.63) and update accordingly the path flows that originate at itself.

5.38 *Optimal Dynamic Routing Based on Window Ratio Strategies.* Consider the optimal routing problem of Section 5.4 in a datagram network. Suppose that we want to implement a set of path flows $\{x_p\}$ calculated on the basis of a nominal set of traffic inputs $\{r_w\}$. The usual solution, discussed in Section 5.5, is to route the traffic of each OD pair w in a way that matches closely the actual fractions of packets routed on the paths $p \in P_w$ with the desired fractions x_p/r_w. With this type of implementation each OD pair w takes into account changes in its own input traffic r_w, but makes no effort to adapt to changes in the input traffic of other OD pairs or to queue lengths inside the network. This problem considers the following more dynamic alternative.

Suppose that each OD pair w calculates the average number of packets N_p traveling on each path $p \in P_w$ by means of Little's theorem

$$N_p = x_p T_p, \qquad \text{for all } p \in P_w, w \in W$$

where T_p is the estimated average round-trip packet delay on path p (time between introduction of the packet into the network and return of an end-to-end acknowledgment for the packet). The origin of each OD pair w monitors the *actual* number of routed but unacknowledged packets $\tilde{N}_p$ on path p and routes packets in a way that roughly equalizes the ratios $\tilde{N}_p/N_p$ for all paths $p \in P_w$. Note that this scheme is sensitive to changes in delay on the paths of an OD pair due to statistical fluctuations of the input traffic and/or the routing policies of other OD pairs. The difficulty with this scheme is that the delays T_p are unknown when the numbers N_p are calculated, and therefore T_p must be estimated somehow. The simplest possibility is to use the measured value of average delay during a preceding time period. You are asked to complete the argument in the following analysis of this scheme for a simple special case.

Suppose that there is a single OD pair and only two paths with flows and path delays denoted x_1, x_2 and $T_1(x)$, $T_2(x)$, respectively, where $x = (x_1, x_2)$. The form of the path delay functions $T_1(x)$, $T_2(x)$ is unknown. We assume that $x_1 + x_2 = r$ for some constant r. Suppose that we measure x_1, x_2, $T_1(x)$, $T_2(x)$, then calculate new nominal path flows $\overline{x}_1$, $\overline{x}_2$ using some unspecified algorithm, and then implement them according to the ratio scheme described above.

(a) Argue that the actual path flow vector $\tilde{x} = (\tilde{x}_1, \tilde{x}_2)$ is determined from $\tilde{x}_1 + \tilde{x}_2 = r$ and the relation

$$\frac{\tilde{x}_1 T_1(\tilde{x})}{\overline{x}_1 T_1(x)} = \frac{\tilde{x}_2 T_2(\tilde{x})}{\overline{x}_2 T_2(x)}$$

(b) Assume that T_1 and T_2 are monotonically increasing in the sense that if $z = (z_1, z_2)$ and $z' = (z'_1, z'_2)$ are such that $z_1 > z'_1$, $z_2 < z'_2$ and $z_1 + z_2 = z'_1 + z'_2$, then $T_1(z) > T_1(z')$ and $T_2(z) < T_2(z')$. Show that the path flows $\tilde{x}_1$, $\tilde{x}_2$ of part (a) lie in the interval between x_1 and $\overline{x}_1$ and x_2 and $\overline{x}_2$, respectively.

(c) Consider a convex cost function of the form $\sum_{(i,j)} D_{ij}(F_{ij})$. Under the assumption in part (b) show that if $\overline{x}$ gives a lower cost than x, the same is true for $\tilde{x}$.

5.39 *Routing in Networks with Frequently Changing Topology [GaB81], [BeT89].* In some situations (*e.g.*, mobile packet radio networks) topological changes are so frequent that topology broadcast algorithms are somewhat impractical. The algorithms of this problem are designed to cope with situations of this type. Consider a connected undirected graph with a special node referred to as the *destination*. Consider the collection C of directed acyclic graphs obtained by assigning a unique direction to each of the undirected links. A graph G in C is said to be *rooted at the destination* if for every node there is a directed path in G starting at the node and ending at the destination. Show that the following two distributed asynchronous algorithms will yield a graph in C that is rooted at the destination starting from any other graph in C.

Algorithm A. A node other than the destination with no outgoing links reverses the direction of all its adjacent links. (This is done repeatedly until every node other than the destination has an outgoing link.)

Algorithm B. Every node i other than the destination keeps a list of its neighboring nodes j that have reversed the direction of the corresponding links (i, j). At each iteration, each node i that has no outgoing link reverses the directions of the links (i, j), for all j that do not appear on its list, and empties the list. If no such j exists (*i.e.*, the list is full), node i reverses the directions of all incoming links and empties the list. Initially, all lists are empty.

Hint: For algorithm A, assign to each node a distinct number. With each set of node numbers, associate the graph in C where each link is directed from a higher to a lower number. Consider an algorithm where each node reverses link directions by changing its number. Use a similar idea for algorithm B.

5.40 *Verifying the Termination of the Distributed Bellman–Ford Algorithm [DiS80], [BeT89].* The objective of the following algorithm (based on the Bellman–Ford method) is to compute in a distributed way a shortest path from a single origin (node 1) to all other nodes *and* to notify node 1 that the computation has terminated. Assume that all links (i, j) are bidirectional, have nonnegative length d_{ij}, maintain the order of messages sent on them, and operate with no errors or failures. Each node i maintains an estimate D_i of the shortest distance from node 1 to itself. Initially, $D_i = \infty$ for all nodes $i \neq 1$, and $D_1 = 0$. Node 1 initiates the computation by sending the estimate $D_1 + d_{1j}$ to all neighbor nodes j. The algorithmic rules for each node $j \neq 1$ are as follows:

1. When node j receives an estimate $D_i + d_{ij}$ from some other node i, after some unspecified finite delay (but before processing a subsequently received estimate), it does the following:
   - **(a)** If $D_j \leq D_i + d_{ij}$, node j sends an ACK to node i.
   - **(b)** If $D_j > D_i + d_{ij}$, node j sets $D_j = D_i + d_{ij}$, marks node i as its best current predecessor on a shortest path, sends an ACK to its previous best predecessor (if any), and sends the estimate $D_j + d_{jk}$ to each neighbor k.
2. Node j sends an ACK to its best current predecessor once it receives an ACK for each of the latest estimates sent to its neighbors.

We assume that each ACK is uniquely associated with a previously sent estimate, and that node 1 responds with an ACK to any length estimate it receives.

(a) Show that eventually node 1 will receive an ACK from each of its neighbors, and at that time D_i will be the correct shortest distance for each node i.

(b) What are the advantages and disadvantages of this algorithm compared with the distributed, asynchronous Bellman–Ford algorithm of Section 5.2.4?