DIMACS

Center for Discrete Mathematics &
Theoretical Computer Science

DIMACS EDUCATIONAL MODULE SERIES

MODULE 03-5
Communications Network Design[1]

Date prepared: October 17, 2003

Steven Cosares
Hofstra University, Hempstead, New York 11549

Fred Rispoli
Dowling College, Oakdale, New York 11769

DIMACS Center, CoRE Bldg., Rutgers University, 96 Frelinghuysen Road, Piscataway, NJ 08854-8018
TEL: 732-445-5928 • FAX: 732-445-5932 • EMAIL: `center@dimacs.rutgers.edu`
Web: `http://dimacs.rutgers.edu/`

Founded as a National Science Foundation Science and Technology Center and a Joint Project of Rutgers University, Princeton University, AT&T Labs - Research, Bell Labs, NEC Laboratories America and Telcordia Technologies with affiliated members Avaya Labs, HP Labs, IBM Research, Microsoft Research.

[1]Research supported by NSF Grant DUE 97-52776

Module Description Information

- **Title:**

  Communications Network Design

- **Author(s):**

  Steven Cosares, Hofstra University, acsstc@hofstra.edu

  Fred Rispoli, Dowling College, rispolif@dowling.edu

- **Abstract:**

  The module introduces students to a version of the Network Design problem (NDP) that arises in the planning for communication networks. A fixed cost is usually associated with placing or activating the link between some pair of nodes. The variable cost associated with a link is proportional to the number of connections supported by the link. The objective in NDP is to find a spanning tree for which the total of the fixed and variable costs is minimized.

- **Informal Description:**

  The module covers some of the basic ideas motivating the use of heuristic solution techniques when an optimal solution is difficult to find. It uses the relatively simple solution techniques for the shortest paths and minimum spanning tree problems as frameworks for generating solutions to this complex problem.

- **Target Audience:**

  The module is intended for undergraduate or graduate students taking a course involving either Discrete Mathematics, Network Flows, Integer Programming, or Optimization.

- **Prerequisites:**

  The module is designed for students that have been introduced to some basic concepts of networks, including the Shortest Paths problem and the Minimum Spanning Tree problem.

- **Mathematical Field:**

  Discrete Mathematics, Network Flows, Algorithms, Mathematical Programming

- **Applications Areas:**

  Telecommunications, Networking, Transportation

- **Mathematics Subject Classification:**

  Primary 90C35; Secondary 65K05

- **Contact Information:**

  Steven Cosares
  Hofstra University, Hempstead, New York 11549

  email: Steven.T.Cosares@Hofstra.edu

- **Other DIMACS modules related to this module:**

  None

TABLE OF CONTENTS

ABSTRACT

The following module introduces students to a version of the Network Design problem (NDP) that arises in the planning for communication networks. The objective in NDP is to find a sub-network in a given graph for which the total of some associated fixed and variable costs is minimized. A fixed cost is usually associated with placing or activating the link between some pair of nodes. The variable cost associated with a link is proportional to the number of connections supported by the link. In this module we focus on a case of NDP in which all traffic originates at some central location, so the underlying sub-network will form a spanning tree.

The module is intended for undergraduate or graduate students taking a course involving Network Flows, Integer Programming, or Optimization. It is designed for a student that has been introduced to some basic concepts of networks, including the Shortest Paths problem and the Minimum Spanning Tree problem. To solve the problem described in this module requires the student to find a spanning tree that is a compromise between the optimal solutions of these two problems. Even this relatively simple version of the Network Design problem is NP-Complete, so the focus of the module is to present easy-to-implement heuristic solution methods. Instructors are encouraged to have advanced students establish some heuristic approaches of their own. A spreadsheet model is also provided with code that allows students to evaluate the cost of their solution proposals. In addition, we show how small problem instances can be solved to optimality using widely available spreadsheet software. These spreadsheets can be used to test the effectiveness of the heuristics proposed.

The module is designed to support a variety of teaching methods, e.g., individual, cooperative, or lab-based. A single example is followed throughout the discussion. Each section in the module ends with an exercise based on the example, which requires the student to immediately apply the concepts presented. We recommend that these exercises be performed with the instructor as part of the classroom experience. Additional examples that can be assigned as homework or lab exercises are also provided.

The Telecommunications division of Megabucks Finance Corporation wants to build a private communications network that connects its corporate headquarters (HQ) with its branch offices around the city. Each link in this network is a physical passageway between a pair of locations, (above or below the ground), over which cables are routed. The network must have an arrangement of links to connect HQ with each of the branch offices using dedicated fiber cables. To establish a link requires activities such as digging trenches, running wires through duct-work, applying for permits to dig into the street, and/or reserving space on telephone poles or in conduits. A local telephone company or cable service provider is usually called upon to perform these tasks. Some links in the network cost more to establish than others, depending on the physical location, the terrain to be built upon, and whether some infrastructure already exists. Once established, a link can hold any number of cables. The total cost of the system depends on which links are established for the network and the total amount of cable and equipment that has to be placed to make the desired connections, which often depends on the lengths of the routes used.

Megabucks wishes to minimize the total cost of the system, so it must find routes for the cables that are relatively short in total distance, while avoiding expensive links wherever possible.

1 Introduction

The situation above is an instance of what is formally referred to as the *Network Design* problem. In a Network Design problem (NDP), one is given an undirected graph where each node represents a key location and each edge represents an eligible location for a network link. Transmission paths, (e.g., fiber-optic cables), are required to connect various pairs of nodes. These cables can only be routed across links that have been selected for inclusion into the network. The relatively simple situation for Megabucks Corp. is represented by the graph in the following figure. The nodes represent corporate headquarters (HQ) and the branch locations. The edges represent viable links for the network. In this instance, a cable connection is required between HQ and each of the other five nodes.

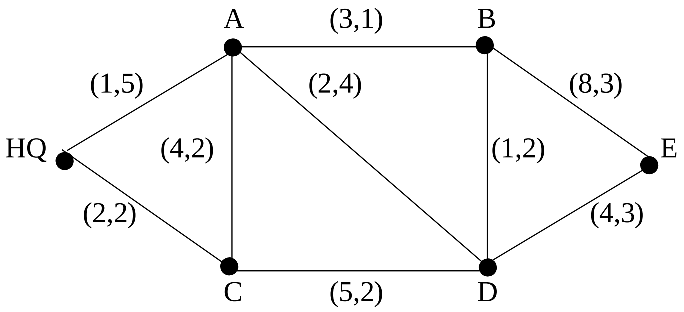


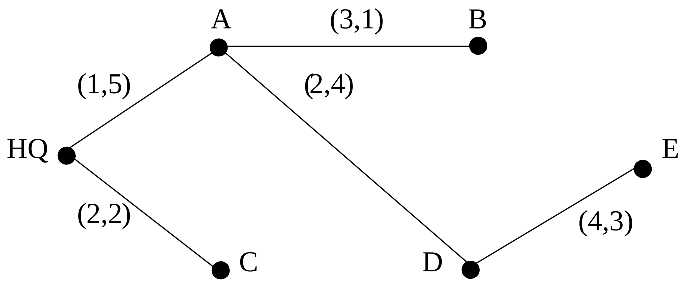
Figure 1

The edges in the graph have labels for the (fixed, variable) costs. The *fixed cost* on the edge between node i and node j, denoted by $F_{ij} (= F_{ji})$, represents the cost of establishing the link between these locations. The *variable cost* on the edge, denoted by $C_{ij} (= C_{ji})$, represents the cost incurred by each transmission cable that uses the link in its route. For Megabucks, it represents the cost of placing a cable along the link and equipping it with repeaters and optical/electrical converters as needed. If the link between nodes i and j is not established, then the associated fixed cost is not incurred. Otherwise, if X transmission cables pass through the link, then the cost incurred at the edge is

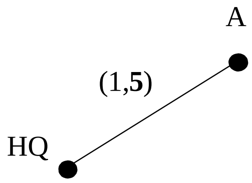
$$F_{ij} + C_{ij} X.$$

The objective in NDP is to minimize the total fixed and variable costs while providing the desired connections. A *feasible solution* to the problem consists of a cable route for each desired connection. The associated *network design* refers to the set of links in the union of the edges traversed by these routes. In an attempt to minimize the costs, a common design requirement is to insist that the connections be supported using as few links as possible. This is accomplished if the union of the cable routes forms a spanning tree of the network, i.e., a minimal set of edges that connect together all of the nodes without forming any cycles. In fact, for the Megabucks problem, where all of the desired cable connections originate at headquarters, the optimal network design must be a spanning tree. (This point is demonstrated in an exercise at the end of the module).

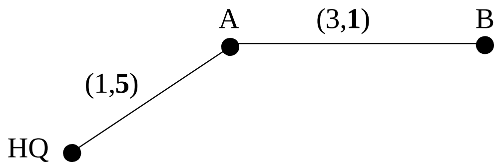
For the network in Figure 1, one possible spanning tree design looks as follows.



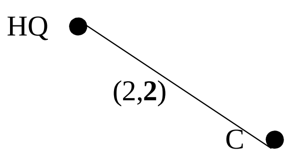
The total of the fixed costs, \$12, is obtained by summing up the left values from each edge in this sub-network. The desired cable connections in the network design must be routed using only these edges. Recall that for a spanning tree, there is exactly one choice for the route between any pair of nodes. The variable costs associated with the cable routes originating at HQ and terminating at each of the other nodes are calculated by summing the right values from the edges in each path, as illustrated below.



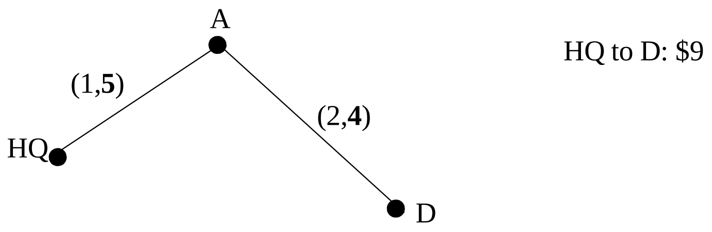
HQ to A: \$5

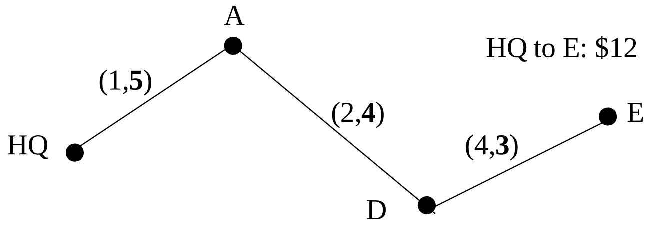


HQ to B: \$6



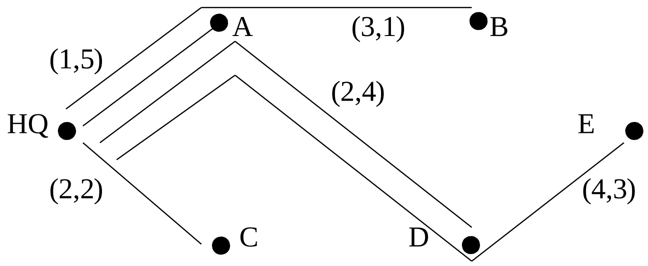
HQ to C: \$2





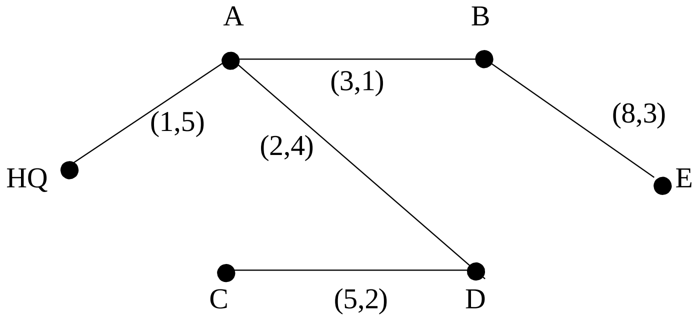
The total of the variable costs is \$5 + \$6 + \$2 + \$9 + \$12 = \$34. Adding the fixed costs, we see that the solution has a total cost of \$34 + \$12 = \$46.

Alternatively, the total cost calculation can be performed on a per edge basis as follows:



Four cables pass through (HQ, A), so the cost incurred at the edge is: $1 + 5\times4 =$ \$21.
One cable passes through (HQ, C), so the cost incurred at the edge is: $2 + 2\times1 =$ \$4.
One cable passes through (A, B), so the cost incurred at the edge is: $3 + 1\times1 =$ \$4.
Two cables pass through (A, D), so the cost incurred at the edge is: $2 + 4\times2 =$ \$10.
One cable passes through (D, E), so the cost incurred at the edge is: $4 + 3\times1 =$ \$7.
So the total cost of the solution is \$46.

Exercise: Find the total of the fixed and variable costs associated with the following spanning tree design.



2 Network Design Problem Notation

Let n represent the number of nodes in the network. One of these nodes is designated as a *root* node. For the network in Figure 1, HQ must be the root node because all of the desired cable routes originate there. For more general problem instances any node can serve as the root.

Let T represent a subset of $n - 1$ edges that comprise a spanning tree for the network. Henceforth, we will only consider network design solutions of this form. For some non-root node j, we define the predecessor of node j, denoted $pred(j)$, to be the node that precedes j in the unique path in T from root node to node j. The root node has no predecessor. Using this notation, one can specify the edges in a tree as a sequence of predecessor nodes. For example, the spanning tree for the network design solution shown in the figure above can be represented as follows:

Node:	HQ	A	B	C	D	E
Predecessor:	∅	HQ	A	HQ	A	D

For a tree T, let $F(T)$ to be the total of the fixed costs from the edges in T.

For each cable connection desired, there is a unique route that uses the edges in T. Let $X(i, j)$ denote the number of times edge (i, j) is used by some route. Clearly $X(i, j) = 0$ for edges that are not in T. Let $V(T)$ be the total of the variable costs incurred by these cable routes, which is equal to

$$\sum_j C_{pred(j),j} X(pred(j), j).$$

Using this notation, we say that the Network Design problem is equivalent to finding a spanning tree T such that $F(T) + V(T)$ is minimized.

3 Heuristic Solution Techniques

Finding a spanning tree having the smallest total fixed and variable costs is a complex task. The tree that minimizes the fixed costs may have large variable costs associated with it; the tree that minimizes the variable costs may use edges with large fixed costs. To find the optimal solution would require a method that mediates between these two objectives. At present, there is no efficient method known that finds the optimal tree, even for this special case where all of the desired cable routes originate at the root node. Any method that is guaranteed to find the optimal solution requires some form of enumeration, e.g., checking the total cost associated with every possible spanning tree. For most communications networks, this approach is unacceptable. The small network shown in Figure 1 has over 45 possible trees. For larger networks, the number of possible spanning trees could be in the hundreds of thousands or in the millions! It would be impossible to examine every possibility within a reasonable amount of time.

As an alternative to expending the effort to find an optimal tree, it may be acceptable to find a less- than-optimal solution that we believe to be relatively cheap. For example, it is reasonable to settle on a solution that is known to cost only a small percentage more than the best possible, if it would take too much effort and time to find a solution that is appreciably better. A *heuristic* method for solving a complex problem, like this one, is defined to be an efficient and reasonable approach to determining a feasible solution. Heuristics methods are usually used to find quality feasible solutions to a problem in situations where it is considered too costly, in time and effort, to search for the optimal solution.

As the name implies, a heuristic method exploits some thoughtful analysis of the problem to suggest a reasonable solution. As observations about the problem are made and special characteristics of the problem are discovered, some solution methods may make themselves apparent.

For example, one could identify some special cases of the Network Design problem for which it is relatively easy to find an optimal tree. Then the method that solves this special case can be used as a heuristic method for the more general cases. Another approach is to find a different problem that has many of the same characteristics as the Network Design problem. Then the successful methods that are used for this problem may be modified to generate good heuristic methods for Network Design. Some examples of these approaches are provided in the following sections. The reader is encouraged to expand and modify this list.

3.1 Applying a Minimum Spanning Tree Algorithm

Observation: If an instance of the Network Design problem is such that the variable costs are all 0, then the optimal network design is the set of edges that comprise a Minimum Spanning Tree (MST) over the fixed costs.

Since the only costs incurred in such an instance of the network design problem are the fixed costs associated with establishing the links, there can be no cheaper solution than the spanning tree with the minimum total cost, i.e., the MST. There is reason to believe that this solution would also be cost effective for instances of the Network Design problem where the variable costs are very small, relative to the fixed costs.

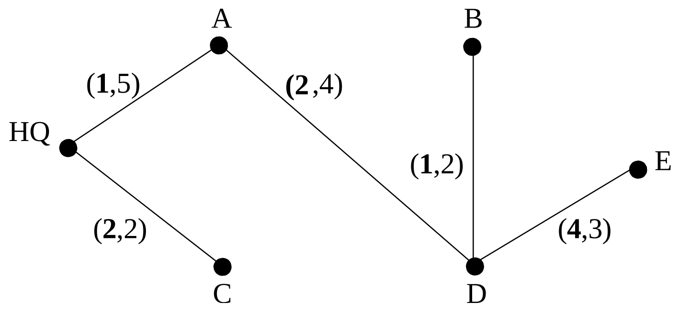
Finding the Minimum Spanning Tree in a graph can be accomplished using any of a number of efficient algorithms. (Consult your class textbook for details.) An implementation of *Prim's algorithm* for finding a MST is presented below. The nodes in the network are labeled $1, \ldots, n$, where node 1 is the root node. The input to the algorithm is a symmetric $n \times n$ matrix of edge weights, called $Weight$. Invalid edges are given a very large weight value. To use Prim's algorithm as a heuristic for Network Design, we assign appropriate values to the weight matrix, e.g., set $Weight(i, j)$ equal to F_{ij}. The output is the set of predecessor nodes that represents the spanning tree of minimum total weight.

Prim's Algorithm

```
Set S = {1};  Set T = {2,...n};
Set pred(1) to 0;
For  j = 2 to n  Set pred(j) to 1;
For  step = 1 to n-1  do
    Let j* = arg min {Weight(pred(j),j) | j in T}
    Set S to S + {j*}; Set T to T - (j*};
    For all i in T do
        If Weight(j*,i) < Weight(pred(i),i)
            Then Set pred(i) to j*;
```

Example 1: Find the Minimum Spanning Tree for the network given in Figure 1, using the fixed costs as edge weights. Calculate the total cost of the network design associated with this solution.

Solution: The following MST is found using Prim's algorithm with $Weight(i, j) = F_{ij}$. The total of the fixed costs is \$1 + \$2 + \$2 + \$1 + \$4 = \$10.



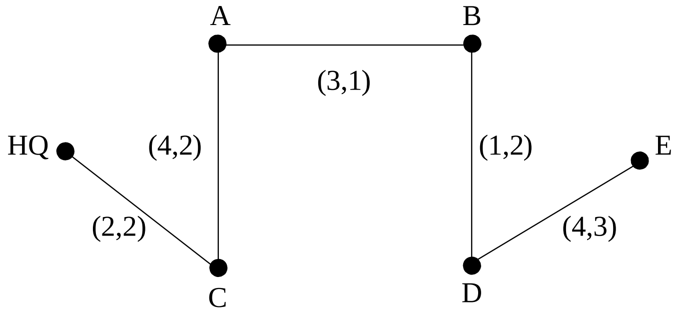
The variable costs are calculated as follows. The route from HQ to node A costs \$5; the route from HQ to node B costs \$5 + \$4 + \$2 = \$11; the route to node C costs \$2; the route to node D costs \$3; and the route to node E costs \$12.

The sum of the fixed and variable costs associated with this solution is \$49. In this case, the solution is not very cost effective because the variable costs are not small, relative to the fixed costs, so perhaps they shouldn't be ignored by the heuristic.

A modification to the heuristic, which may produce a better solution, is to set the weight matrix to incorporate the variable costs as well as the fixed costs. For example, set $Weight(i,j)$ equal to $F_{ij} + C_{ij}$ and run the Prim algorithm for that matrix, or set $Weight(i,j)$ to $F_{ij} + 2C_{ij}$ and run the Prim algorithm. More generally, one can set $Weight(i,j)$ equal to $F_{ij} + \alpha C_{ij}$ for various positive values of α, run the Prim algorithm for each, evaluate the associated total cost, and select the best of the network designs found. Since the Prim algorithm can be implemented to run very quickly, a large number of scenarios can be examined.

Example 2: Find the Minimum Spanning Tree for the network given in Figure 1, using the matrix $Weight(i,j) = F_{ij} + 2C_{ij}$. Calculate the total cost of the network design associated with this solution.

Solution: The following tree is found using Prim's algorithm with those weights. The total of the fixed costs of the edges from the tree is \$2 + \$4 + \$3 + \$1 + \$4 = \$14



The route from HQ to A has a total variable cost of \$4; the route from HQ to B costs \$5; HQ to C costs \$2; HQ to D costs \$7; and the route from HQ to E costs \$10. The total cost of the design is \$42, which is less than the cost of the design found by considering only the fixed costs for link selections.

Exercise: Find the Minimum Spanning Tree for the network given in Figure 1, using the matrix $Weight(i,j) = F_{ij} + 3C_{ij}$. Calculate the total cost of the network design associated with this solution.

3.2 Applying a Shortest Paths Algorithm

Dijkstra's algorithm is a commonly used method to find the route of shortest total length between a pair of nodes in a network. Consider the shortest route between root node 1 and node 2, the shortest route between node 1 and node 3, ..., and the shortest route between node 1 and node n. The union of these $n-1$ routes can be reduced to a set of edges that comprise a spanning tree for the network. We define this tree to be the *Shortest Paths Tree* (SPT) rooted at node 1. As we shall see, Dijkstra's algorithm can be used to find a SPT.

Observation: If an instance of the Single-source Network Design problem is such that the fixed costs are all 0, then the optimal network design is the set of edges that comprise the Shortest Paths Tree rooted at the source node, using the variable costs as edge lengths.

Problem instances with no fixed costs arise when the only costs are associated with placing the cables. This happens if the network has infrastructure that is already in place. In such cases, finding the shortest route for each desired connection, with the variable costs serving as edge lengths, solves the Network Design problem. Hence, there is reason to believe that for instances where the fixed costs are small, relative to the variable costs, the SPT provides a cost-effective solution to the Network Design problem as well.

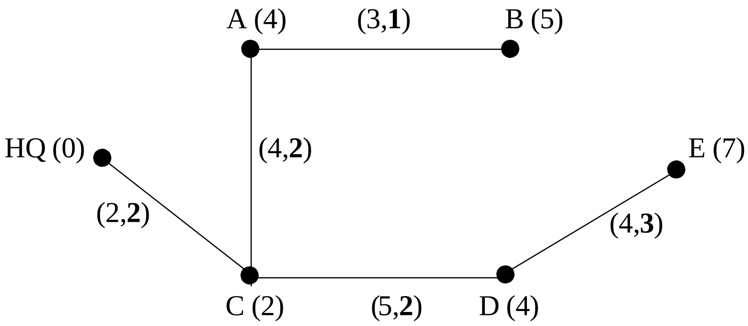
An implementation of Dijkstra's algorithm to find an SPT is provided below. The nodes in the network are labeled $1, \ldots, n$, where node 1 is the root node. The input to Dijkstra's algorithm is a non-negative $n \times n$ matrix of edge lengths, called *Length*. Invalid edges are given a very large length value. To use Dijkstra's algorithm as a heuristic for Network Design, we assign appropriate values to the length matrix, e.g., set $Length(i, j)$ equal to C_{ij}. The output is the set of predecessor nodes that represents a spanning tree that minimizes the total of the lengths of the routes from the root node.

Dijkstra's Algorithm

```
Set S = {1};  Set T = {2,...n};
Set pred(1) to 0; Set path(1) to 0;
For  j = 2 to n  Set pred(j) to 1;
For  step = 1 to n-1  do
    Let j* = arg min {path(pred(j))+ Length(pred(j),j) | j in T}
    Set S to S + {j*}; Set T to T - (j*};
    Set path(j*) to path(pred(j*))+ Length(pred(j*),j*)
    For all i in T do
        If path(j*)+ Length(j*,i) < path(pred(i))+ Length(pred(i),i)
            Then Set pred(i) to j*;
```

Example 3: Find the Shortest Paths Tree for the network given in Figure 1, using the variable costs as edge lengths. Calculate the total cost of the network design associated with this solution.

Solution: The following SPT is found using Dijkstra's algorithm with $Length(i, j) = C_{ij}$.. Next to each node is placed the sum of the variable costs for the path from HQ.



The total of the variable costs is \$4 + \$5 + \$2 + \$4 + \$7 = \$22, i.e., the sum of the path values at the nodes. The total of the fixed costs is \$2 + \$4 + \$3 + \$5 + \$4 = \$18. So the total cost of the design is \$40.

In many cases this heuristic approach suffers because fixed costs may not be small, relative to the variable costs, so they shouldn't be ignored by the heuristic. A method that addresses this issue is presented in one of the exercises.

Two important points arise:

1) The smallest possible value for the total of the fixed costs is found by using Prim's algorithm, (\$14 in this example), and the smallest possible value for the total of the variable costs is found by using Dijkstra's algorithm, (\$22 in this example). Therefore, we know that the optimal network design cannot have a total cost that is less than the sum of these two values, (\$36 in this case). This value can be used to judge the quality of the solution obtained by any heuristic.

2) For most networks, as in our example, the Shortest Paths Tree and the Minimum Spanning Tree are usually comprised of a different set of edges, even if they are based on the same set of edge weights / lengths. If an instance of network design is such that the SPT based on the variable costs and the MST based on the fixed costs results in the same tree, then this tree must be the optimal network design because it would achieve the lower bound noted above.

3.3 A Local Improvement Heuristic

The term *Local Improvement* refers to a solution method where one starts with some reasonable (but not necessarily optimal) solution to a problem and then checks to see if some small change to the solution would result in a lower total cost. For some problem types, this method ultimately finds the optimal solution. For such problems, when no improvements are possible, the solution can be shown to be optimal.

The challenge for more complex problems, like Network Design, is that there is no known local improvement method that is guaranteed to terminate with an optimal solution. That is, it may be possible to have a solution for which no marginal improvement is possible, yet the solution is far from optimal. In such cases, Local Improvement methods are used as heuristic tools to obtain a *locally* optimal solution rather than a *global* optimum.

The following Local Improvement algorithm, which is often referred to as a *Network Simplex* method, finds the optimal solution to the Shortest Paths problem. Since the problem is similar to the Network Design problem with a single source, there is reason to believe it will be an effective heuristic.

Step 0: Start with some spanning tree of the network. Call this the Present Tree.

Step 1: Consider placing a new arc (i,j) in the solution, (where j is not in the route from the root node to node i). To maintain the tree structure, $(pred(j),j)$ must be removed from the solution. Call this new tree the Candidate Tree.

Step 2: If the Candidate Tree has a smaller total cost than the Present Tree, set the Candidate Tree as the new Present Tree.

Step 3: If no new arc provides an improvement over the Present Tree

Then STOP. The Present Tree is the solution.

Otherwise GOTO Step 1.

Appendix 1 describes a set of Excel spreadsheets that can be used to apply this approach manually. Code is provided that reads in a proposed tree solution and quickly calculates the total of the associated fixed and variable costs. A new arc is placed in the tree by changing the predecessor of one of the nodes. The code then evaluates the cost of the new tree. The code also indicates when a proposed solution is not feasible, i.e., when the specified set of edges does not form a spanning tree.

A more sophisticated implementation of Local Improvement, which we call the *Shortcut Method*, is described below. In this heuristic, a calculation is made that determines in advance whether some arc (i,j) would provide an improvement over the Present Tree T.

For the nodes in tree T, we require two labels:

$Clabel(j)$ is set to the sum of the variable costs in the unique route in T between the root node and node j.

$X(j)$ is set to the number of cable routes crossing the edge $(pred(j),j)$ in T

A *shortcut* arc is an edge $(i,j) \notin T$, where j is not in the route from the root to node i and:

$$F_{ij} + (Clabel(i) + C_{ij})X(j) < F_{pred(j),j} + Clabel(j)X(j).$$

Including a shortcut arc would result in cost savings over Present Tree T. The Shortcut heuristic is as follows:

Step 0: Start with some spanning tree of the network; call this the Present Tree. For each node j, find the values for $X(j)$ and $Clabel(j)$.

Step 1: Repeat the following steps until no shortcut arcs are found:

If (i,j) is a shortcut arc place (i,j) in the Present Tree and remove $(pred(j),j)$. Set $pred(j)$ to i.

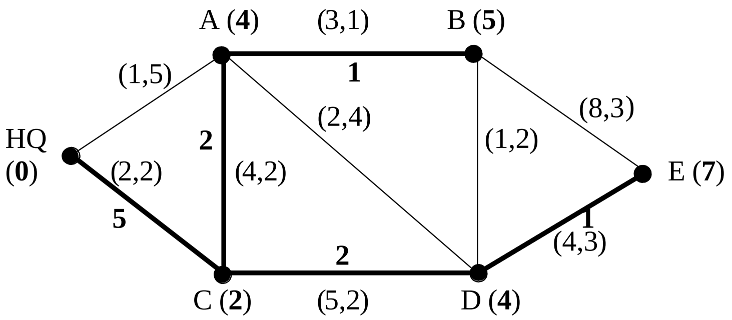
Recalculate the values of $Clabel()$ and $X()$ for this new tree.

We point out that during each iteration, edge (i,j) may be examined twice: once as the arc from node i to node j, and once as the arc from node j to node i. It is possible for edge (i,j) to be a shortcut in either direction.

In order to implement this algorithm on a computer, the programmer will have to make some choices. First, he or she will have to determine which spanning tree to start with for Step 0. The method visits a sequence of tree solutions until it finds one in which no immediate improvement is possible. In a typical instance of NDP, there are many such solutions. Starting this process with two different trees would probably generate two different sequences, so may lead to two different final solutions (or local optima). We recommend running the algorithm for a variety of starting trees, then selecting the best of the final solutions encountered.

Example 4: Using the SPT solution as a starting tree, determine the values of $Clabel(j)$ and $X(j)$ for every node j. List all of the shortcut arcs based on this solution.

Solution: For each node j, the value for $Clabel(j)$ is placed in bold next to node j. The value for $X(j)$ is placed in bold on edge $(pred(j), j)$.



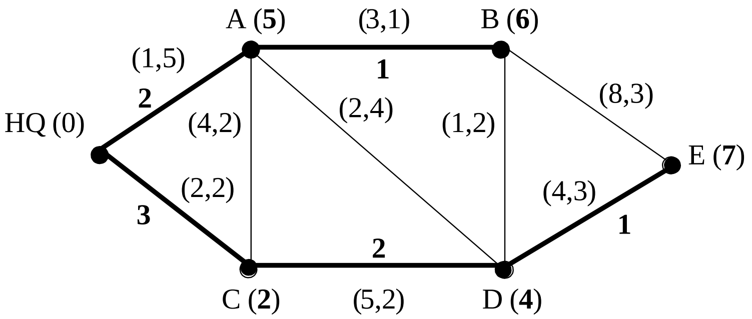
	$F_{ij} + (Clabel(i) + C_{ij})X(j)$	$F_{pred(j),j} + Clabel(j)X(j)$	Shortcut?
(HQ,A):	$1 + (0 + 5) \times 2 = 11$	$4 + 4 \times 2 = 12$	Yes
(A, D):	$2 + (4 + 4) \times 2 = 18$	$5 + 4 \times 2 = 13$	No
(B, D):	$1 + (5 + 2) \times 2 = 15$	$5 + 4 \times 2 = 13$	No
(B, E):	$8 + (5 + 3) \times 1 = 16$	$4 + 7 \times 1 = 11$	No
(D, A):	$2 + (4 + 4) \times 2 = 18$	$4 + 4 \times 2 = 12$	No
(D, B):	$1 + (4 + 2) \times 1 = 7$	$3 + 5 \times 1 = 8$	Yes
(E, B):	$8 + (7 + 3) \times 1 = 18$	$3 + 5 \times 1 = 8$	No

The edge (HQ, A) has a fixed cost of 1 and a variable cost of 5. It is a shortcut arc because $(1 + (0 + 5) \times 2) < (4 + (4 \times 2))$. So including the edge and removing edge (C,A) results in a cost savings of \$1. The edge (D,B) is also a shortcut arc.

The second choice a programmer is faced with is to determine which of the shortcut arcs, if there are more than one, to add to the Present Tree solution. One approach is to use the first shortcut arc encountered. Another approach is to use the shortcut arc with the greatest difference in the inequality; this is the arc that provides the greatest immediate benefit. The first has the advantage of requiring less work per iteration; the second provides for greater descent in total cost per iteration. For a variety of reasons, neither approach is guaranteed to find a better final solution than the other one.

Example 5: Give the new design associated with adding edge (HQ,A). Update $Clabel()$ and $X()$ for this new solution.

Solution: This new design has a total cost of \$39:

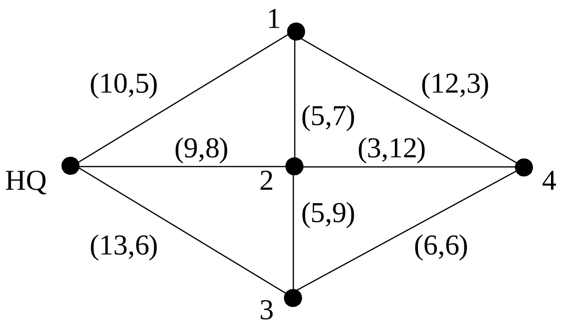


We emphasize that the solution obtained by the Shortcut heuristic may not be optimal. Moreover, since starting with different initial trees, or examining the potential shortcut arcs in different orders, a variety of solutions can be obtained. None of these are guaranteed to have a cost that is "good", i.e., within some range of the optimal. The only known way to obtain a solution with such guarantees is to apply some form of enumeration of the possible spanning trees. On the other hand, the solution obtained through this heuristic has the property that there are no obvious shortcuts. This condition may be an acceptable criterion for our network design needs. To capitalize on this approach, the final solution could be obtained by applying the heuristic to a fixed number of distinct starting trees, run some implementation of the Shortcut heuristic for each case, and then choosing the particular tree that has the smallest associated costs.

Exercise: Determine whether there are any shortcut arcs in the solution from Example 5. If there are make the appropriate improvement to the solution. Repeat the process until no immediate improvement is possible.

Exercises

1) Consider the network with the following (fixed, variable) costs:



Determine which of the following three spanning trees yields the smallest total cost.

a)

Node:	HQ	1	2	3	4
Predecessor:	Ø	HQ	HQ	HQ	1

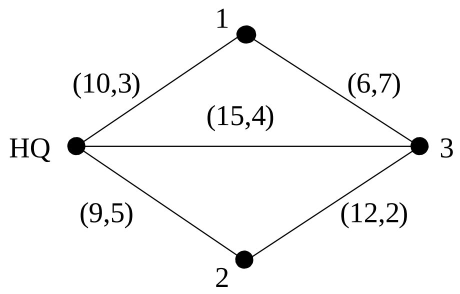
b)

Node:	HQ	1	2	3	4
Predecessor:	Ø	HQ	HQ	HQ	2

c)

Node:	HQ	1	2	3	4
Predecessor:	Ø	HQ	HQ	HQ	3

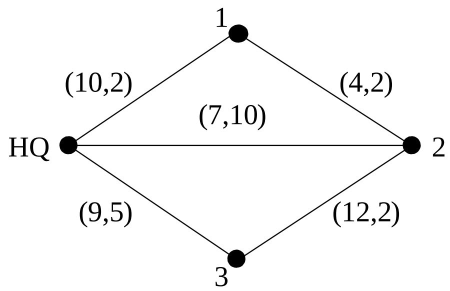
2) Consider the network with the following (fixed, variable) costs:



a) List all eight spanning trees for this network.
b) Find the optimal network design from among these possible trees.

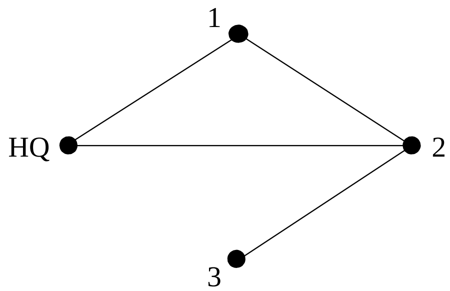
3) Determine the number of different spanning trees for the graph given in Exercise 1.

4) Consider the following network.



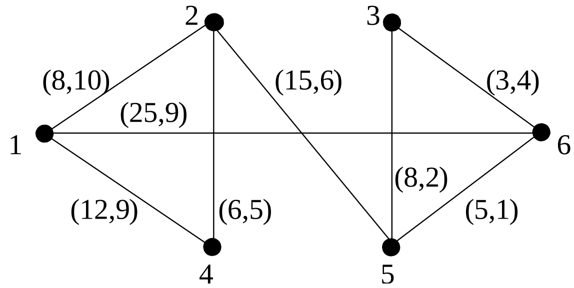
a) Evaluate the total fixed and variable costs associated with the solution specified by the routes: HQ to 1, HQ to 1 to 2, and HQ to 2 to 3.

b) The feasible solution given in part a) uses the following links.



Notice that this network is not a spanning tree. Use this example to demonstrate that within any non- tree feasible solution, there is a spanning tree feasible solution whose total cost is no greater.

5) Consider the network with the following (fixed, variable) costs:



a) Compute the total fixed and variable costs of the following tree design.

Node:	1	2	3	4	5	6
Predecessor:	Ø	1	6	1	2	1

b) Find the values of *Clabel* and X for each of the nodes in the tree.

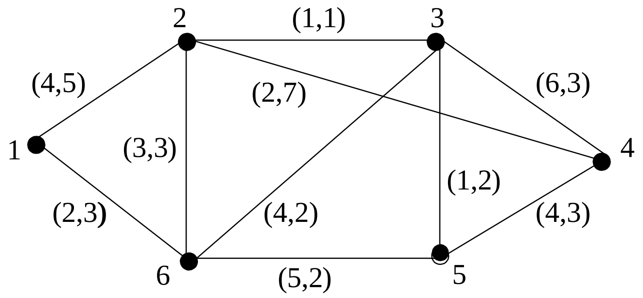
c) Using the values for *Clabel* and X determine whether $(5,3)$ is a shortcut arc.

d) Using the values for *Clabel* and X determine whether $(3,5)$ is a shortcut arc.

e) Show that arc $(5,6)$ is a shortcut arc.

f) Give the new solution when arc $(5,6)$ replaces arc $(1,6)$ in the spanning tree. Determine the improvement in total cost.

6) For the network with the following (fixed, variable) costs:



a) Calculate the total fixed and variable costs of the following spanning tree design:

Node:	1	2	3	4	5	6
Predecessor:	Ø	6	6	3	3	1

b) Use Prim's algorithm to find a Minimum Spanning Tree using the fixed costs as edge weights. Calculate the total fixed and variable cost associated with this design solution.

c) Use Dijkstra's algorithm to find a Shortest Paths Tree rooted at node 1, using the variable costs as edge lengths. Calculate the total cost associated with this solution.

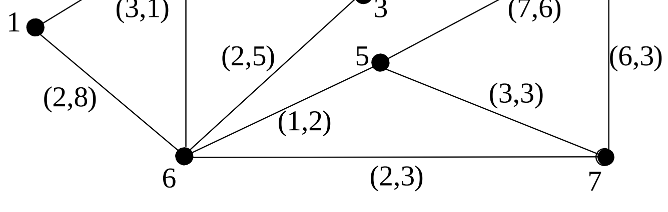
d) Use your answers to parts b) and c) to determine a lower bound to the total cost of an optimal solution.

e) Use Prim's algorithm to find a Minimum Spanning Tree using the sum of the fixed and variable costs as edge weights. Calculate the total cost associated with this design solution.

f) Using the best of the solutions found in parts a) to e), find the value of *Clabel* and X for each of the nodes. Determine whether there are any cost improving shortcut arcs. If there are, make the appropriate change to the network design solution.

g) Continually repeat part f), performing the Local Improvement heuristic until you arrive at a solution for which there are no shortcut arcs.

7) For the network with the following (fixed, variable) costs:



a) Calculate the total fixed and variable costs of following spanning tree design:

Node:	1	2	3	4	5	6	7
Predecessor:	Ø	1	6	3	4	1	5

b) Use Prim's algorithm to find a Minimum Spanning Tree using the fixed costs as edge weights. Calculate the total fixed and variable costs associated with this design solution.

c) Use Dijkstra's algorithm to find a Shortest Paths Tree rooted at node 1, using the variable costs as edge lengths. Calculate the total cost associated with this solution.

d) Use Prim's algorithm to find a Minimum Spanning Tree using the sum of the fixed and variable costs as edge weights. Calculate the total cost associated with this design solution.

e) Use your answers to parts b) and c) to determine a lower bound to the total cost of an optimal solution.

f) Using the best of the solutions found in parts a) to d), find the value of Clabel and X for each of the nodes. Determine whether there are any cost improving shortcut arcs. If there are, make the appropriate change to the network design solution.

g) Continually repeat part f), performing the Local Improvement heuristic until you arrive at a solution for which there are no shortcut arcs.

8) A modification to the heuristic that applies Dijkstra's algorithm to find a tree solution is to set the length matrix to reflect the fixed costs as well as the variable costs. For example, set $Length(i, j) = C_{ij} + F_{ij}$ and run the Dijkstra's algorithm for that matrix, or set $Length(i, j) = C_{ij} + \frac{1}{2}F_{ij}$ and run Dijkstra's algorithm.

More generally, one can set $Length(i, j) = C_{ij} + \alpha F_{ij}$ for various values of α between 0 and 1, run Dijkstra's algorithm for each, determine the total cost associated, and then select the best of the network designs found. (Note: The value of α should probably be small since only a fraction of the fixed cost is incurred on a per-connection basis).

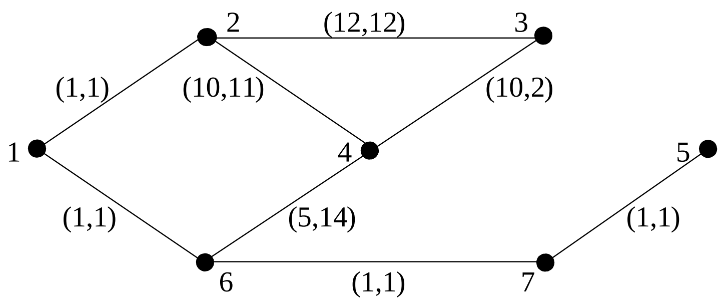
a) For the network in Figure 1, use Dijkstra's algorithm to find the Shortest Paths Tree using the matrix $Length(i, j) = C_{ij} + F_{ij}$. Calculate the total cost of the network design associated with this solution.

b) Find the Shortest Paths Tree using the matrix $Length(i, j) = C_{ij} + \frac{1}{8}F_{ij}$. Calculate the total cost of the network design associated with this solution.

c) Repeat parts (a) and (b) using the network given in Exercise 6.

d) Repeat parts (a) and (b) using the network given in Exercise 7.

9) Consider the network with the following (fixed, variable) costs:



a) Show that there are no shortcut arcs for the spanning tree given by

Node:	1	2	3	4	5	6	7
Predecessor:	Ø	1	4	2	7	1	6

i.e., that arcs $(2,3)$, $(3,2)$, $(4,6)$, and $(6,4)$ are not shortcut arcs.

b) Show that the absence of shortcut arcs is not a sufficient condition for an optimal solution by finding a different spanning tree whose total cost is lower than the spanning tree given in part (a). (*Hint*: Consider an interchange that adds two arcs and drops two arcs at the same time.)

Appendix 1: Network Design Evaluation using Spreadsheets

Spreadsheet software is a popular tool that is used in business and personal computing to perform basic storage and analysis of quantitative data. Because such software nowadays has powerful capabilities and is easy to use, spreadsheets are also used as teaching tools to effectively describe quantitative problems and various approaches to their solution. In this section we show how spreadsheets can be used to quickly evaluate the cost of a proposed tree design solution, even for fairly large problem instances. Such functionality could be used to implement the Local Improvement heuristic method by allowing the user to visit a large number of solution proposals before selecting a final design.

A spreadsheet-based file is typically called a workbook, which consists of a set of related worksheets, each containing text, data, functions and formulas. A workbook also has access to some computer programs (e.g., written in Visual Basic for Applications) that can be used to manipulate the data in the worksheets. For Network Design Evaluation, we provide three sheets[2], called "Fixed Costs", "Variable Costs", and "Evaluator", and a program called "Calculate". The worksheets containing the cost data are formatted as the following example illustrates.

"Fixed Costs"

	A	B	C	D	E	F	G	H
1	Node	1	2	3	4	5	6	
2	1	10000	1	10000	2	10000	10000	
3	2	1	10000	3	4	2	10000	
4	3	10000	3	10000	10000	1	8	
5	4	2	4	10000	10000	5	10000	
6	5	10000	2	1	5	10000	4	
7	6	10000	10000	8	10000	4	10000	
8								

"Variable Costs"

	A	B	C	D	E	F	G	H
1	Node	1	2	3	4	5	6	
2	1	10000	5	10000	2	10000	10000	
3	2	5	10000	1	2	4	10000	
4	3	10000	1	10000	10000	2	3	
5	4	2	2	10000	10000		10000	
6	5	10000	4	2		10000	3	
7	6	10000	10000	3	10000	3	10000	
8								

The first row and column of the worksheets are reserved for node labels to make data entry easy. The cost for edge (i,j) is located in the $(i+1)^{\text{st}}$ row and the $(j+1)^{\text{st}}$ column in the worksheet. Ineligible edges are represented with a sufficiently large cost value.

[2]This Excel workbook is available from the authors.

The “Evaluator” worksheet contains the data associated with the size of the network and the proposed solution. A button on the worksheet evokes the program “Calculate”, which retrieves the data and places the costs on the sheet. A change to the tree is implemented by changing the predecessor of one of the nodes. The code also indicates when a proposed solution is not feasible, i.e., when the proposed set of edges does not form a spanning tree. A particularly large total cost would indicate that some ineligible edge is used in the solution. The format of the “Evaluator” worksheet and code for the “Calculate” program are provided below.

Tree Solution Evaluation:						
# Nodes:	6	Evaluate		Total Fixed Costs	13	
				Total Variable Costs	18	
Trial Solution:						
Node	Predecessor			Total Cost:	31	
1	0					
2	1					
3	5					
4	1					
5	4					
6	5					

```
Sub Calculate()
  Dim pred(50), X(50) As Integer
  Sheets("Evaluator").Cells(7, 3) = " "
  n = Sheets("Evaluator").Cells(3, 2)
  For i = 1 To n
      pred(i) = Sheets("Evaluator").Cells(6 + i, 2).Value
      X(i) = 0
  Next i
  For i = 2 To n
      X(i) = X(i) + 1
      prev = pred(i)
      While Not (prev = 1)
          X(prev) = X(prev) + 1
          If (X(prev) > n) Then
          prev = 1
          Sheets("Evaluator").Cells(7, 3) = "  NOT A TREE!!!!"
          Else
          prev = pred(prev)
          End If
      Wend
  Next i
  fixtot = 0
  vartot = 0
  For i = 2 To n
      fixtot = fixtot + Sheets("Fixed Costs").Cells(1 + pred(i), 1 + i)
      vartot = vartot + X(i) * Sheets("Variable Costs").Cells(1 + pred(i), 1 + i)
```

```
  Next i
  Sheets("Evaluator").Cells(3, 6) = fixtot
  Sheets("Evaluator").Cells(4, 6) = vartot
  Sheets("Evaluator").Cells(6, 6) = fixtot + vartot
End Sub
```

Appendix 2: Solving the Network Design Problem as an Integer Program

Additional approaches to solving the Network Design problem require formulating it as an Integer Programming problem. The following notation will be used in the formulation, which represents edge (i, j) in the network as two directed arcs (i, j) and (j, i):

$n =$ the number of nodes

$F_{ij} =$ the fixed cost of arc (i,j). Note that $F_{ij} = F_{ji}$.

$C_{ij} =$ the variable cost of arc (i,j). Note that $C_{ij} = C_{ji}$.

Let y_{ij} be the binary decision variable that is set to 1 if arc (i, j) is included in the solution and set to 0 otherwise. Let x_{ij} represent the number of cable routes from node 1 (the root node) that are routed across arc (i, j). Note that if x_{ij} is positive, then y_{ij} must be 1.

The formulation for the single-source Network Design problem is as follows:

(1) Minimize $\sum_{i=1}^{n}\sum_{j=2}^{n} C_{ij}x_{ij} + \sum_{i=1}^{n}\sum_{j=2}^{n} F_{ij}y_{ij}$

Subject to:

(2) $\sum_{p=1}^{n} x_{pj} - \sum_{q=2}^{n} x_{jq} = 1,\ j = 2, \ldots, n$

(3) $My_{ij} \geq x_{ij} \qquad i = 1, \ldots, n; \qquad j = 2, \ldots, n$

(4) $y_{ij} \in \{0, 1\} \qquad i = 1, \ldots, n; \qquad j = 2, \ldots, n$

(5) $x_{ij} \geq 0, \text{integer} \qquad i = 1, \ldots, n; \qquad j = 2, \ldots, n$

Notice that the objective function (1) consists of two terms. The first term accounts for the total variable cost, and the second accounts for the total fixed costs. Constraint set (2) represents the "flow-balance" constraints. It ensures that exactly one transmission path terminates at each node j. Constraint set (3) is called the edging constraint set and ensures that if y_{ij} is 0, then no cable route will use arc (i, j), i.e., x_{ij} would have to be 0. M is some large number that allows x_{ij} to be unconstrained if y_{ij} is 1.

Using this formulation, the network design problem can now be solved using some general Integer Programming solution tool. This means that it would be possible to obtain some solutions without having to invest the time and effort to implement approaches that are specific to the Network Design problem. However, one should not expect this approach to be effective for larger problem instances, e.g., networks with a large number of nodes or networks that are somewhat dense.

Finding Optimal Solutions with a Spreadsheet Solver

For moderately sized networks or networks that are sparse, one can obtain optimal Network Design solutions by using the Integer Programming formulation and mathematical programming software.

For example, Solver is an optimization program that is included in many spreadsheet software packages, such as Lotus and Excel. It has been used to solve many well-known network flow problems and can be applied to Network Design as well. Currently, the standard version of Solver that is included in many spreadsheet software packages can be used to obtain optimal solutions for the network design problem with networks containing 200 arcs or less.

To solve a Network Design problem with Solver, we set up a worksheet as indicated in the following diagram. This example solves the problem given in Figure 1. Rows 4 to 19 represents the eligible directed arcs in the network. Columns A to I represent the input data, i.e. the cost and connectivity information. The path variables x_{ij} are represented in column K and the binary decision variables y_{ij} are represented in column L. Column M represents the slack in the edging constraints. Row 20 contains formulas that calculate the number of paths from HQ that terminate at each of the nodes. For this problem, we want this value to be one. The total cost of the solution is represented by a formula placed in cell L21.

	A	B	C	D	E	F	G	H	I	J	K	L	M
1													
2			Variable	Fixed								Link	Edging
3	From	To	Costs	Costs	A	B	C	D	E		# Paths	Used?	Constraints
4	HQ	A	5	1	1						0	0	0
5	HQ	C	2	2			1				0	0	0
6	A	C	2	4	-1		1				0	0	0
7	C	A	2	4	1		-1				0	0	0
8	A	B	1	2	-1	1					0	0	0
9	B	A	1	2	1	-1					0	0	0
10	A	D	4	3	-1			1			0	0	0
11	D	A	4	3	1			-1			0	0	0
12	C	D	2	5			-1	1			0	0	0
13	D	C	2	5			1	-1			0	0	0
14	B	D	2	1		-1		1			0	0	0
15	D	B	2	1		1		-1			0	0	0
16	B	E	3	8		-1			1		0	0	0
17	E	B	3	8		1			-1		0	0	0
18	D	E	3	4				-1	1		0	0	0
19	E	D	3	4				1	-1		0	0	0
20					0	0	0	0	0				
21											Total Cost:	0	

The key formulas used in this spreadsheet model, using the Excel program are as follows.

```
E20 =  SUMPRODUCT(E4:E19, K4:K19)
F20 =  SUMPRODUCT(F4:F19, K4:K19)
G20 =  SUMPRODUCT(G4:G19, K4:K19)
H20 =  SUMPRODUCT(H4:H19, K4:K19)
I20 =  SUMPRODUCT(I4:I19, K4:K19)
L21 =  SUMPRODUCT(C4:C19, K4:K19) + SUMPRODUCT(D4:D19, L4:L19)

M4 = K4 - 100 * L4
M5 = K5 - 100 * L5
.
.
.
M19 = K19 - 100 * L19
```

In Excel, Solver can be accessed from the Tools menu. In the Solver dialogue box we must enter the Target Cell which is L21, to be minimized. The Changing Cells are K4:K19 and L4:L19. We must also input the following constraints:

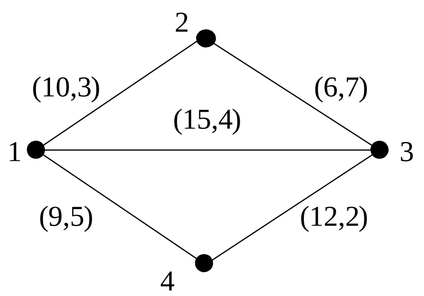
```
E20:I20  =  1
K4:K19 integers
K4:K19  >=  0
L4:L19  binary
M4:M19  <=  0
```

The last step is to access the Solver Options and input Assume Linear Model. All other settings can be left at their default values. The following worksheet gives the optimal solution to the problem in Figure 1. Observe that the optimal tree consists of arcs whose associated binary variable is one; namely arcs from HQ to A, HQ to C, C to D, D to B, and D to E.

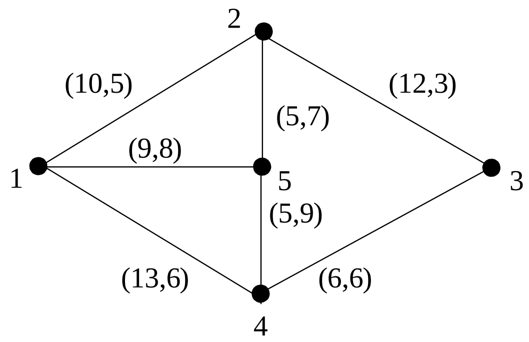
	A	B	C	D	E	F	G	H	I	J	K	L	M	
1														
2			Variable	Fixed								Link	Edging	
3	From	To	Costs	Costs	A	B	C	D	E		# Paths	Used?	Constraints	
4	HQ	A	5	1	1						1	1	-99	
5	HQ	C	2	2			1				4	1	-96	
6	A	C	2	4	-1		1				0	0	0	
7	C	A	2	4	1		-1				0	0	0	
8	A	B	1	2	-1	1					0	0	0	
9	B	A	1	2	1	-1					0	0	0	
10	A	D	4	3	-1			1			0	0	0	
11	D	A	4	3	1			-1			0	0	0	
12	C	D	2	5			-1	1			3	1	-97	
13	D	C	2	5			1	-1			0	0	0	
14	B	D	2	1		-1		1			0	0	0	
15	D	B	2	1		1		-1			1	1	-99	
16	B	E	3	8		-1			1		0	0	0	
17	E	B	3	8		1			-1		0	0	0	
18	D	E	3	4				-1	1		1	1	-99	
19	E	D	3	4				1	-1		0	0	0	
20					1	1	1	1	1					
21											Total Cost:	37		
22														

Spreadsheet Exercises

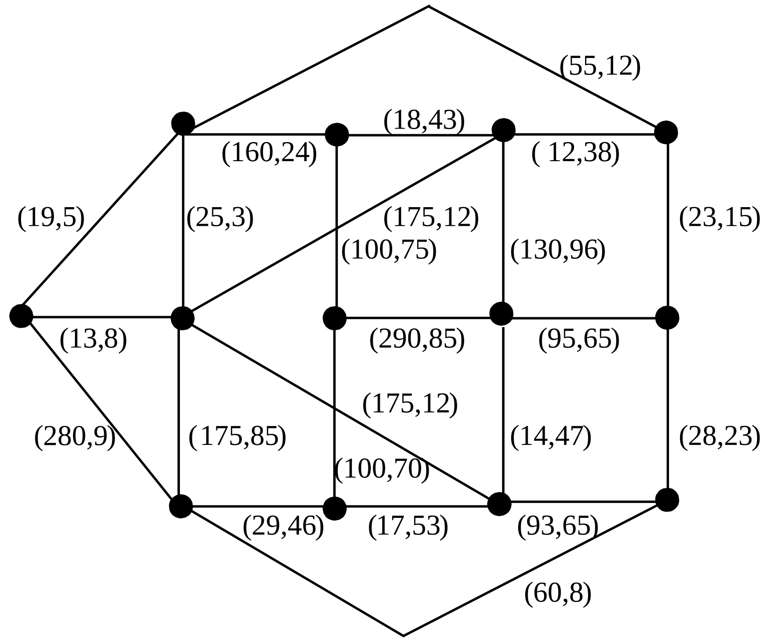
1) Give the mathematical programming formulation for the network design problem specified by the following network. Then use any mathematical programming software such as Solver to obtain an optimal solution.



2) Give the mathematical programming formulation for the network design problem specified by the network below. Then use Solver to obtain an optimal solution.



3) Construct a spreadsheet model similar to the one described in Appendix 1 to find a solution to the communication network design problem for the network given below.



(a) Find any spanning tree of this network. Use the Evaluator spreadsheet to determine the total of the fixed and variable cost associated with your spanning tree design.

(b) Modify your tree by changing the predecessor to some node. Re-calculate the total costs. If your new solution is an improvement, keep it; if not revert to your previous tree. Repeat the process by adjusting another predecessor until you are satisfied with your solution.

(c) Repeat the process, starting with the MST, where $Weight(i, j)$ is set to the value of the fixed cost of the edge. Is your final solution any better?

(d) Repeat the process, starting with the MST, where $Weight(i, j)$ is set to the sum of the fixed and variable costs of the edge.

(e) Repeat the process, starting with the tree that is the SPT, where $Length(i, j)$ is set to the value of the variable cost of the edge.

4) Build the Solver spreadsheet model to find the optimal design for the network in Exercise 3 and find an optimal solution. How well does your heuristic solution obtained in Exercise 3 compare to an optimal solution?

5) A more general version of the Network Design Problem is to design a network where every node, except the root, requires d individual cable connections instead of 1. This type of problem

may be modeled by changing the right hand side of constraint set (2) in the integer programming formulation to d, instead of 1. For example, to find an optimal solution to this variation of the problem for the Network given in Figure 1, one could use the same worksheet described in Appendix 2, and keep all inputs to the Solver dialogue box the same, except set cells E20:I20 = d, instead of E20:I20 = 1.

(a) Use Solver to solve this version of the Network Design Problem for the network given in Exercise 3, when d = 2, 5 and 10.

(b) Given a spanning tree, the average path length is the average of the lengths of each of the $n-1$ paths from the root to each node. Compare the average path length of the optimal solutions obtained in part (a) to the average path length of the optimal solution obtained in Exercise 4. Why do you suppose it changes?

6) a) Explain how the spreadsheet model given in Appendix 2 can be used to find a minimum spanning tree for the network given in Figure 1.

(b) Use Solver to find a minimum spanning tree for the network given in Exercise 3 using the fixed costs as weights. If the maximum time is exceeded before an optimal solution is found, try increasing the maximum time (e.g., use 600 seconds.) Check your answer by finding a minimum spanning tree using Prim's algorithm.

References

Some general references for network optimization problems are the following.

Ahuja, R., T. Magnanti, and J. Orlin, *Network Flows*, Prentice Hall, Englewood Cliffs, 1993.

Cook, W., W. Cunningham, W. Pulleyblank and A. Schrijver, *Combinatorial Optimization*, Wiley, New York, 1998.

Ford, L. and D. Fulkerson, *Flows in Networks*, Princeton U. Press, 1962.

Hu, T. and M. Shing, *Combinatorial Algorithms and Network Algebra*, Dover Press, New York, 2002.

Jungnickel, D., *Graphs, Networks, and Algorithms*, Springer-Verlag, Berlin, 1999.

If you want to learn more about the Network Design problem and additional heuristic approaches, you are encouraged to read the following publications. Various versions of the problem are presented. They also contain comprehensive lists of references to research on the problem.

M. Minoux, "Network Synthesis and Optimum Network Design Problems: Models, Solution Methods and Applications", *Networks*, Vol. 19 pp. 313-360 (1989).

Magnanti, Wolsey, and Wong, "Network Design", *Handbooks in Operations Research and Management Science*, Vol. 6: Networks, North-Holland, Amsterdam (1992).

The single-source version of the problem is discussed in the following paper. The paper discusses the complexity of the problem and presents a set of heuristics that exploit the Mathematical Programming formulation of the problem.

D. Hochbaum and A. Segev, "Analysis of a Flow Problem with Fixed Charges", *Networks*, Vol. 19 pp. 291-312 (1989).

The multiple source-sink version of the Network Design problem, which arises in typical telecommunications network applications, is addressed in the following papers. The problem is treated as a Multi-commodity Flow problem with fixed costs. A "dual-based" approach allows for some decomposition that allows for the solution of large instances.

Balakrishnan, Magnanti, and Wong, "A Dual-Ascent Procedure for Large-Scale, Uncapacitated Network Design", *Operations Research*, Vol. 37 pp.716-740 (1989).

Balakrishnan, Magnanti, Shulman, and Wong, "Models for capacity expansion in local access telecommunications networks", *Annals of Operations Research*, Vol. 33 pp.239-284 (1991).

The use of spreadsheets for decision modeling is described with a multitude of examples in the following publications.

R. Hesse, *Managerial Spreadsheet Modeling and Analysis*, Irwin, Chicago IL, (1997) .

C. Ragsdale, "Spreadsheet Modeling and Decision Analysis: A Practical Introduction to Management Science", Course Technology, Cambridge MA, (1995).