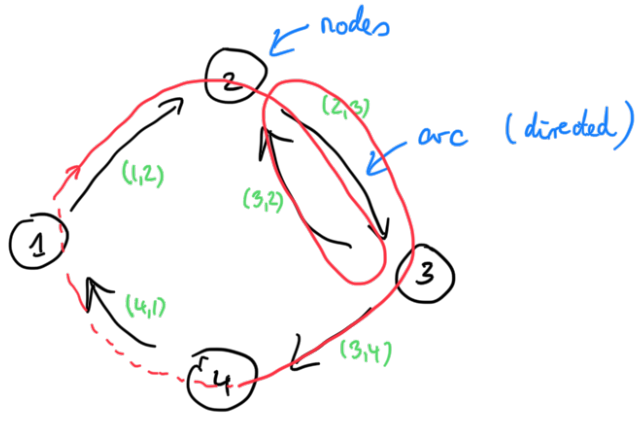
Network optimization problems



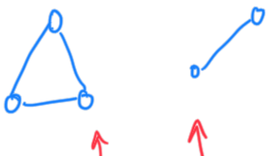
(1,2), (2,3), (3,2), (2,3), (3,4), $\underbrace{(4,1)}_{\text{makes path a cycle}}$

CS/Math	OR		
vertices	nodes	element of a set	N
edges	arcs	(i,j)	, i,j ∈ N
graph	network		

path: sequence of matching arcs (i,j), (j,k), (k,m), ... (can be directed or undirected)

cycle: path where starting point and endpoint are the same

A network is <u>connected</u> if there is an undirected path between any two nodes.

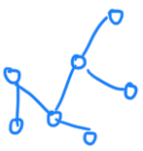


not connected

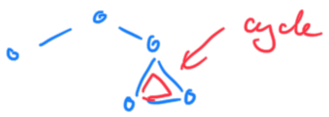
no path between these nodes!

A <u>tree</u> is a network which is connected, but doesn't have cycles

a tree:



not a tree:



n- shortest path algorithm to solve the shortest path problem:

n	node	distance	
1	A	2	
2	B	4	(via O-A-B)
	or C	4	(via O-C)
3	E	7	via (O-A-B-E)
4	D	8	via (O-A-B-E-D)
5	T	13	via (O-A-B-D)

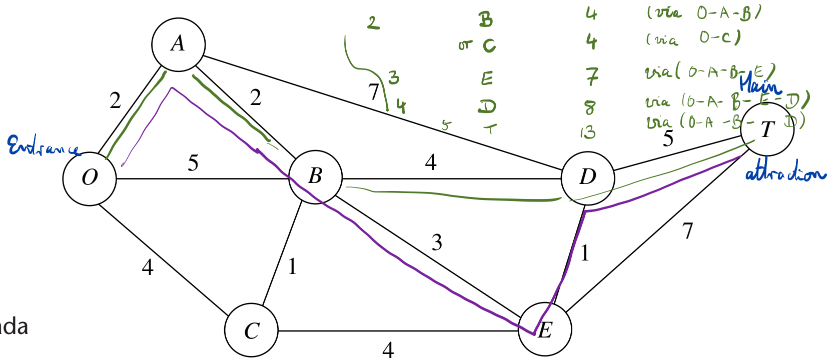


FIGURE 9.1
The road system for Seervada Park.

"n shortest paths algorithm": For n = 1, 2, 3, ...
• keep a list of the n nodes with the shortest connection to O,
• terminate if T is in list

will be installed under just enough roads to provide some connection between every pair of stations. The question is where the lines should be laid to accomplish this with a *minimum* total number of miles of line installed. (This is an example of the minimum spanning tree problem to be discussed in Sec. 9.4.)

The third problem is that more people want to take the tram ride from the park entrance to station *T* than can be accommodated during the peak season. To avoid unduly disturbing the ecology and wildlife of the region, a strict ration has been placed on the number of tram trips that can be made on each of the roads per day. (These limits differ for the different roads, as we shall describe in detail in Sec. 9.5.) Therefore, during the peak season, various routes might be followed regardless of distance to increase the number of tram trips that can be made each day. The question pertains to how to route the various trips to *maximize* the number of trips that can be made per day without violating the limits on any individual road. (This is an example of the maximum flow problem to be discussed in Sec. 9.5.)

9.2 THE TERMINOLOGY OF NETWORKS

A relatively extensive terminology has been developed to describe the various kinds of networks and their components. Although we have avoided as much of this special vocabulary as we could, we still need to introduce a considerable number of terms for use throughout the chapter. We suggest that you read through this section once at the outset to understand the definitions and then plan to return to refresh your memory as the terms are used in subsequent sections. To assist you, each term is highlighted in **boldface** at the point where it is defined.

A network consists of a set of *points* and a set of *lines* connecting certain pairs of the points. The points are called **nodes** (or vertices); e.g., the network in Fig. 9.1 has seven nodes designated by the seven circles. The lines are called **arcs** (or links or edges or branches); e.g., the network in Fig. 9.1 has 12 arcs corresponding to the 12 roads in the road system. Arcs are labeled by naming the nodes at either end; for example, *AB* is the arc between nodes *A* and *B* in Fig. 9.1.

Algorithm for the Minimum Spanning Tree Problem.

1. Select any node arbitrarily, and then connect it (i.e., add a link) to the nearest distinct node.
2. Identify the unconnected node that is closest to a connected node, and then connect these two nodes (i.e., add a link between them). Repeat this step until all nodes have been connected.
3. Tie breaking: Ties for the nearest distinct node (step 1) or the closest unconnected node (step 2) may be broken arbitrarily, and the algorithm must still yield an optimal solution. However, such ties are a signal that there may be (but need not be) multiple optimal solutions. All such optimal solutions can be identified by pursuing all ways of breaking ties to their conclusion.

The fastest way of executing this algorithm manually is the graphical approach illustrated next.

Applying This Algorithm to the Seervada Park Minimum Spanning Tree Problem

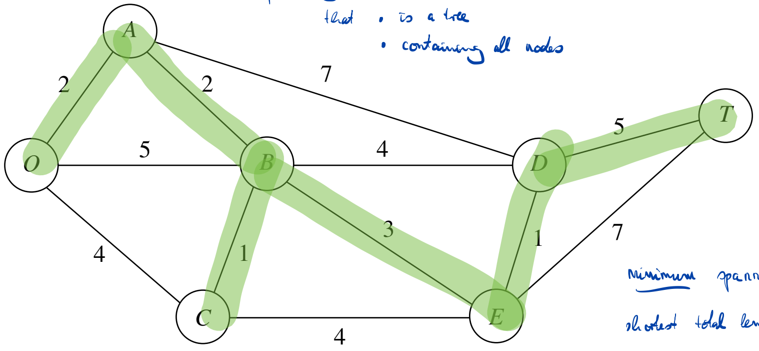
The Seervada Park management (see Sec. 9.1) needs to determine under which roads telephone lines should be installed to connect all stations with a minimum total length of line. Using the data given in Fig. 9.1, we outline the step-by-step solution of this problem.

Nodes and distances for the problem are summarized below, where the thin lines now represent *potential* links.

spanning tree: sub-network obtained by deleting arcs
that • is a tree
• containing all nodes

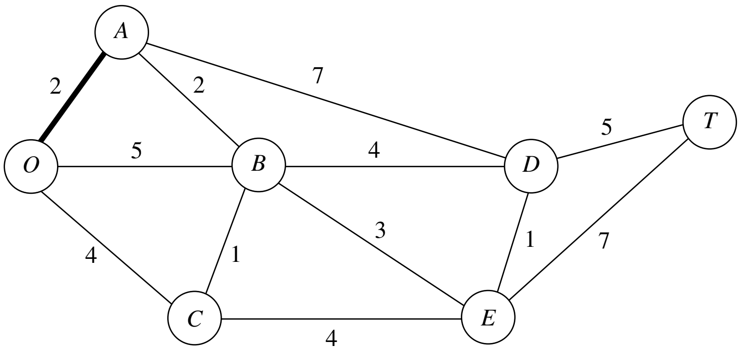
Minimum spanning tree:

• a greedy algorithm will be optimal!



minimum spanning tree: shortest total length of arcs.

Arbitrarily select node O to start. The unconnected node closest to node O is node A. Connect node A to node O.



Solving the max flow problem via the "augmented path algorithm":

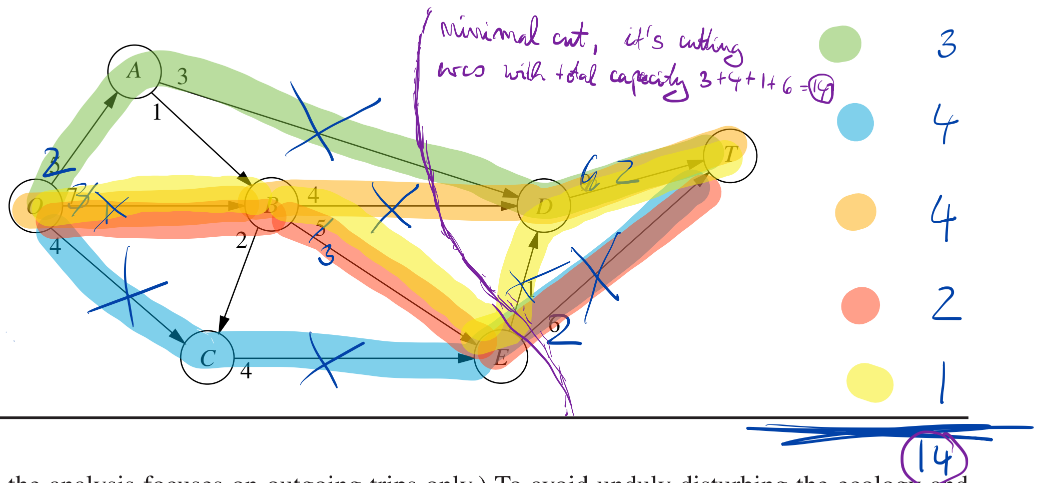

minimal cut, it's cutting arcs with total capacity 3+4+1+6 = 14
A
B
C
D
E
O
T
3
1
4
2
5
4
1
6
2
4
3
2
2
3
4
4
2
1
14

FIGURE 9.6
The Seervada Park maximum flow problem.

the analysis focuses on outgoing trips only.) To avoid unduly disturbing the ecology and wildlife of the region, strict upper limits have been imposed on the number of outgoing trips allowed per day in the outbound direction on each individual road. For each road, the direction of travel for outgoing trips is indicated by an arrow in Fig. 9.6. The number at the base of the arrow gives the upper limit on the number of outgoing trips allowed per day. Given the limits, one *feasible solution* is to send 7 trams per day, with 5 using the route $O \rightarrow B \rightarrow E \rightarrow T$, 1 using $O \rightarrow B \rightarrow C \rightarrow E \rightarrow T$, and 1 using $O \rightarrow B \rightarrow C \rightarrow E \rightarrow D \rightarrow T$. However, because this solution blocks the use of any routes starting with $O \rightarrow C$ (because the $E \rightarrow T$ and $E \rightarrow D$ capacities are fully used), it is easy to find better feasible solutions. Many *combinations* of routes (and the number of trips to assign to each one) need to be considered to find the one(s) maximizing the number of trips made per day. This kind of problem is called a *maximum flow problem.*

In general terms, the maximum flow problem can be described as follows.

1. All flow through a directed and connected network originates at one node, called the **source,** and terminates at one other node, called the **sink.** (The source and sink in the Seervada Park problem are the park entrance at node O and the scenic wonder at node T, respectively.)
2. All the remaining nodes are *transshipment nodes.* (These are nodes A, B, C, D, and E in the Seervada Park problem.)
3. Flow through an arc is allowed only in the direction indicated by the arrowhead, where the maximum amount of flow is given by the *capacity* of that arc. At the *source,* all arcs point away from the node. At the *sink,* all arcs point into the node.
4. The objective is to maximize the total amount of flow from the source to the sink. This amount is measured in either of two equivalent ways, namely, either the amount *leaving the source* or the amount *entering the sink.*

Some Applications

Here are some typical kinds of applications of the maximum flow problem.

1. Maximize the flow through a company's distribution network from its factories to its customers.
2. Maximize the flow through a company's supply network from its vendors to its factories.

DC → W2, and F1 → W1 → W2) for shipping to W2. Factory F2 has just one route to W2 (F2 → DC → W2) and one to W1 (F2 → DC → W2 → W1). The cost per unit shipped through each shipping lane is shown next to the arrow. Also shown next to F1 → F2 and DC → W2 are the maximum amounts that can be shipped through these lanes. The other lanes have sufficient shipping capacity to handle everything these factories can send.

The decision to be made concerns how much to ship through each shipping lane. The objective is to minimize the total shipping cost.

Formulation as a Linear Programming Problem. With seven shipping lanes, we need seven decision variables ($x_{\text{F1-F2}}$, $x_{\text{F1-DC}}$, $x_{\text{F1-W1}}$, $x_{\text{F2-DC}}$, $x_{\text{DC-W2}}$, $x_{\text{W1-W2}}$, $x_{\text{W2-W1}}$) to represent the amounts shipped through the respective lanes.

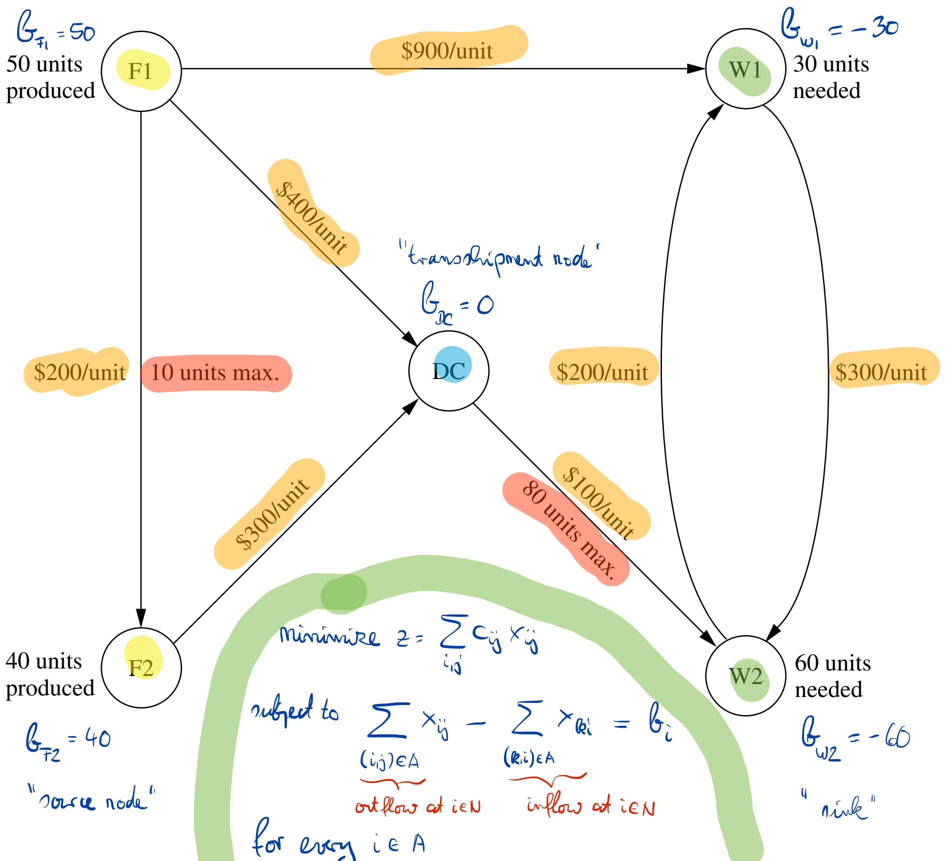
There are several restrictions on the values of these variables. In addition to the usual nonnegativity constraints, there are two *upper-bound constraints,* $x_{\text{F1-F2}} \leq 10$ and $x_{\text{DC-W2}} \leq 80$, imposed by the limited shipping capacities for the two lanes, F1 → F2 and DC → W2. All the other restrictions arise from five *net flow constraints,* one for each of the five locations. These constraints have the following form.

Net flow constraint for each location:

Amount shipped out − amount shipped in = required amount.

As indicated in Fig. 3.13, these required amounts are 50 for F1, 40 for F2, −30 for W1, and −60 for W2.

FIGURE 3.13
The distribution network for Distribution Unlimited Co.



$\boxed{\text{Min cost flow problem}}$

nodes $i \in N$

arcs $(i,j) \in A$, $i,j \in N$

c_{ij} unit cost of transportation on arc (i,j)

u_{ij} max capacity on arc (i,j)

b_i source at node $i \in N$

x_{ij}: decision variable: # of units to be transported from i to j

minimize $z = \sum_{ij} c_{ij} x_{ij}$

subject to $\underbrace{\sum_{(i,j)\in A} x_{ij}}_{\text{outflow at } i \in N} - \underbrace{\sum_{(k,i)\in A} x_{ki}}_{\text{inflow at } i \in N} = b_i$

for every $i \in A$

$0 \leq x_{ij} \leq u_{ij}$ for every $(i,j) \in A$

What is the required amount for DC? All the units produced at the factories are ultimately needed at the warehouses, so any units shipped from the factories to the distribution center should be forwarded to the warehouses. Therefore, the total amount shipped from the distribution center to the warehouses should *equal* the total amount shipped from the factories to the distribution center. In other words, the *difference* of these two shipping amounts (the required amount for the net flow constraint) should be *zero*.

Since the objective is to minimize the total shipping cost, the coefficients for the objective function come directly from the unit shipping costs given in Fig. 3.13. Therefore, by using money units of hundreds of dollars in this objective function, the complete linear programming model is

$$\text{Minimize} \quad Z = 2x_{\text{F1-F2}} + 4x_{\text{F1-DC}} + 9x_{\text{F1-W1}} + 3x_{\text{F2-DC}} + x_{\text{DC-W2}} + 3x_{\text{W1-W2}} + 2x_{\text{W2-W1}},$$

subject to the following constraints:

1. Net flow constraints:

$$\begin{aligned}
x_{\text{F1-F2}} + x_{\text{F1-DC}} + x_{\text{F1-W1}} &= 50 \text{ (factory 1)}\\
-x_{\text{F1-F2}} + x_{\text{F2-DC}} &= 40 \text{ (factory 2)}\\
- x_{\text{F1-DC}} - x_{\text{F2-DC}} + x_{\text{DC-W2}} &= 0 \text{ (distribution center)}\\
- x_{\text{F1-W1}} + x_{\text{W1-W2}} - x_{\text{W2-W1}} &= -30 \text{ (warehouse 1)}\\
- x_{\text{DC-W2}} - x_{\text{W1-W2}} + x_{\text{W2-W1}} &= -60 \text{ (warehouse 2)}
\end{aligned}$$

2. Upper-bound constraints:

$$x_{\text{F1-F2}} \leq 10, \qquad x_{\text{DC-W2}} \leq 80$$

3. Nonnegativity constraints:

$$x_{\text{F1-F2}} \geq 0, \quad x_{\text{F1-DC}} \geq 0, \quad x_{\text{F1-W1}} \geq 0, \quad x_{\text{F2-DC}} \geq 0, \quad x_{\text{DC-W2}} \geq 0, \quad x_{\text{W1-W2}} \geq 0, \quad x_{\text{W2-W1}} \geq 0.$$

You will see this problem again in Sec. 9.6, where we focus on linear programming problems of this type (called the *minimum cost flow problem*). In Sec. 9.7, we will solve for its optimal solution:

$$x_{\text{F1-F2}} = 0, \quad x_{\text{F1-DC}} = 40, \quad x_{\text{F1-W1}} = 10, \quad x_{\text{F2-DC}} = 40, \quad x_{\text{DC-W2}} = 80, \quad x_{\text{W1-W2}} = 0, \quad x_{\text{W2-W1}} = 20.$$

The resulting total shipping cost is \$49,000.

You also will see a case study involving a *much* larger problem of this same type at the end of the next section.

3.5 SOME CASE STUDIES

To give you a better perspective about the great impact linear programming can have, we now present three case studies of *real* applications. Each of these is a *classic* application, initiated in the early 1980s, that has come to be regarded as a standard of excellence for future applications of linear programming. The first one will bear some strong similarities to the Wyndor Glass Co. problem, but on a realistic scale. Similarly, the second and