10.46 **Multicommodity flow models** seek a minimum cost flow where separate commodities are moving through a common network.

EXAMPLE 10.6: BAY FERRY MULTICOMMODITY FLOW

As usual, it will help to think of a simple (fictitious) example. Figure 10.22 depicts traffic flows in communities around an ocean bay. Each morning, the population of the three residential communities travels to the two industrial and two commercial centers in the region. Table 10.3 details flows by origin and destination. For example, 1250 of the 6000 daily trips originating at residential node 4 have industrial park node 7 as their destination.

At present, geography limits each trip to a single path. Numbers on arcs of the network in Figure 10.22 show the distance in kilometers between various points. For instance, those originating at node 1 and bound for node 7 must drive all the way around the bay through nodes 2 to 6. The 21,100 trips arising daily in the three residential communities produce a total of 399,250 kilometers of driving along such routes.

Regional planners are considering various improvements to reduce air pollution by reducing the number of kilometers driven. One idea is the ferry indicated by

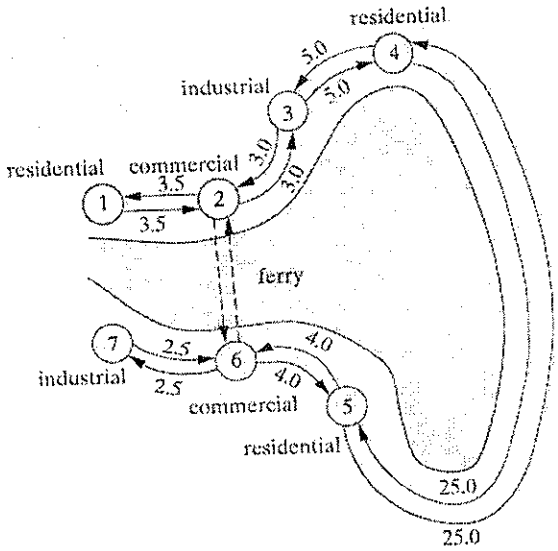


FIGURE 10.22 Bay Ferry Example Network

TABLE 10.3 Daily Trips in the Bay Ferry Example

Trip Origin Node	Total Trips	Trips by Destination						
		1	2	3	4	5	6	7
1	2,850	—	900	750	40	10	600	550
4	6,000	100	2000	1100	—	150	1400	1250
5	12,250	110	4000	2200	200	—	3500	2440

dashed arcs (2, 6) and (6, 2) in Figure 10.22. If a ferry were introduced, it could carry 2000 cars in each direction during the morning rush period. We want to know how many kilometers of driving might be saved.

Multicommodity Flow Models

Clearly, Figure 10.22 depicts a flow network. But what is flowing? If we treat all trips as equal, we have only a single commodity. Then, however, it would be feasible to fulfill the demand for 1250 trips from origin node 4 at sink node 7 with trips from any source. Naturally, a minimum distance solution would prefer ones from nearby source 5, leaving demands for node 5 trips at locations 2 and 3 to be filled from the closer source 1. Such a solution makes no sense for the application because trips are not fungible.

We must form separate commodity networks for trips from each of the three sources. That is, we must model in multiple commodities. Still, the commodities are not independent. All share the 2000-trip capacity of the proposed ferry. Such interdependencies are typical of multicommodity flows.

10.47 Commodities of a multicommodity flow model cannot be analyzed separately because they interact through shared arc capacities.

Suppose that we employ constants

$c_{q,i,j} \triangleq$ unit cost of commodity q flow in arc (i, j)

$u_{i,j} \triangleq$ shared capacity of arc (i, j)

$b_{q,k} \triangleq$ net demand for commodity q at node k

and the decision variables

$x_{q,i,j} \triangleq$ commodity q flow in arc (i, j)

10.48 The multicommodity network flow model on a digraph with nodes $k \in V$ and arcs $(i, j) \in A$ is

$$\min \quad \sum_{q} \sum_{(i,j)\in A} c_{q,i,j} x_{q,i,j}$$

$$\text{s.t.} \quad \sum_{(i,k)\in A} x_{q,i,k} - \sum_{(k,j)\in A} x_{q,k,j} = b_{q,k} \qquad \text{for all } q, k \in V$$

$$\sum_{q} x_{q,i,j} \le u_{i,j} \qquad \text{for all } (i, j) \in A$$

$$x_{q,i,j} \ge 0 \qquad \text{for all } q, (i, j) \in A$$

Table 10.4 presents the corresponding formulation of our Bay Ferry example. There commodity 1 corresponds to flows from origin node 1, commodity 2 denotes flows from residential node 4, and commodity 3 relates to residential node 5. Notice that there are separate systems of flow conservation equations for each commodity, plus a common set of capacity constraints on the two arcs with flow limits.

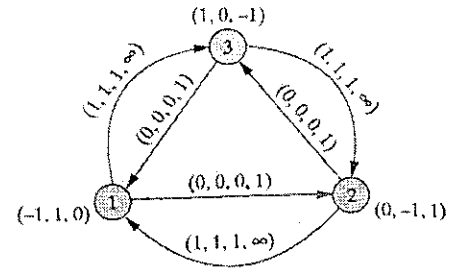
TABLE 10.4 Bay Ferry Example Model

min	$3.5x_{1,1,2} + 3.5x_{1,2,1} + 3x_{1,2,3} + 3x_{1,3,2} + 5x_{1,3,4} + 5x_{1,4,3}$			(minimize driving)
	$+ 15x_{1,4,5} + 15x_{1,5,4} + 4x_{1,5,6} + 4x_{1,6,5} + 2.5x_{1,6,7} + 2.5x_{1,7,6}$			
	$+ 3.5x_{2,1,2} + 3.5x_{2,2,1} + 3x_{2,2,3} + 3x_{2,3,2} + 5x_{2,3,4} + 5x_{2,4,3}$			
	$+ 15x_{2,4,5} + 15x_{2,5,4} + 4x_{2,5,6} + 4x_{2,6,5} + 2.5x_{2,6,7} + 2.5x_{2,7,6}$			
	$+ 3.5x_{3,1,2} + 3.5x_{3,2,1} + 3x_{3,2,3} + 3x_{3,3,2} + 5x_{3,3,4} + 5x_{3,4,3}$			
	$+ 15x_{3,4,5} + 15x_{3,5,4} + 4x_{3,5,6} + 4x_{3,6,5} + 2.5x_{3,6,7} + 2.5x_{3,7,6}$			
s.t.	$x_{1,2,1} - x_{1,1,2}$	=	−2,850	(commodity 1)
	$x_{1,1,2} + x_{1,3,2} + x_{1,6,2} - x_{1,2,1} - x_{1,2,3} - x_{1,2,6}$	=	900	
	$x_{1,2,3} + x_{1,4,3} - x_{1,3,2} - x_{1,3,4}$	=	750	
	$x_{1,3,4} + x_{1,5,4} - x_{1,4,3} - x_{1,4,5}$	=	40	
	$x_{1,4,5} + x_{1,6,5} - x_{1,5,4} - x_{1,5,6}$	=	10	
	$x_{1,2,6} + x_{1,5,6} + x_{1,7,6} + x_{1,6,2} - x_{1,6,5} - x_{1,6,7}$	=	600	
	$x_{1,6,7} - x_{1,7,6}$	=	550	
	$x_{2,2,1} - x_{2,1,2}$	=	100	(commodity 2)
	$x_{2,1,2} + x_{2,3,2} + x_{2,6,2} - x_{2,2,1} - x_{2,2,3} - x_{2,2,6}$	=	2,000	
	$x_{2,2,3} + x_{2,4,3} - x_{2,3,2} - x_{2,3,4}$	=	1,100	
	$x_{2,3,4} + x_{2,5,4} - x_{2,4,3} - x_{2,4,5}$	=	−6,000	
	$x_{2,4,5} + x_{2,6,5} - x_{2,5,4} - x_{2,5,6}$	=	150	
	$x_{2,2,6} + x_{2,5,6} + x_{2,7,6} + x_{2,6,2} - x_{2,6,5} - x_{2,6,7}$	=	1,400	
	$x_{2,6,7} - x_{2,7,6}$	=	1,250	
	$x_{3,2,1} - x_{3,1,2}$	=	110	(commodity 3)
	$x_{3,1,2} + x_{3,3,2} + x_{3,6,2} - x_{3,2,1} - x_{3,2,3} - x_{3,2,6}$	=	4,000	
	$x_{3,2,3} + x_{3,4,3} - x_{3,3,2} - x_{3,3,4}$	=	2,200	
	$x_{3,3,4} + x_{3,5,4} - x_{3,4,3} - x_{3,4,5}$	=	200	
	$x_{3,4,5} + x_{3,6,5} - x_{3,5,4} - x_{3,5,6}$	=	−12,250	
	$x_{3,2,6} + x_{3,5,6} + x_{3,7,6} - x_{3,6,2} - x_{3,6,5} - x_{3,6,7}$	=	3,300	
	$x_{3,6,7} - x_{3,7,6}$	=	2,440	
	$x_{1,2,6} + x_{2,2,6} + x_{3,2,6}$	≤	2,000	(capacities)
	$x_{1,6,2} + x_{2,6,2} + x_{3,6,2}$	≤	2,000	
	all $x_{q,i,j} \ge 0$			

An optimal solution reduces the total driving to 280,770 kilometers, a savings of 29.7%. This is accomplished by accommodating $x_{1,2,6} = 1160$ commodity 1 trips on the 2-to-6 ferry, and $x_{2,2,6} = 840$ commodity 2. In the reverse direction the ferry carries $x_{3,6,2} = 2000$ commodity 3 trips.

SAMPLE EXERCISE 10.29: FORMULATING MULTICOMMODITY FLOWS

Consider the following multicommodity flow problem:



Labels on arcs show costs for three commodities and common capacity $(c_{1,i,j}, c_{2,i,j}, c_{3,i,j}, u_{i,j})$. Labels on nodes show net demands $(b_{1,k}, b_{2,k}, b_{3,k})$. Formulate the corresponding multicommodity network flow model.

Modeling: Following format 10.48, the model is

$$\begin{aligned}
\min \quad & x_{1,1,3} + x_{1,3,2} + x_{1,2,1} + x_{2,1,3} + x_{2,3,2} \\
& + x_{2,2,1} + x_{3,1,3} + x_{3,3,2} + x_{3,2,1} \\
\text{s.t.} \quad & x_{1,2,1} + x_{1,3,1} - x_{1,1,2} - x_{1,1,3} = -1 \\
& x_{1,1,2} + x_{1,3,2} - x_{1,2,1} - x_{1,2,3} = 0 \\
& x_{1,1,3} + x_{1,2,3} - x_{1,3,1} - x_{1,3,2} = 1 \\
& x_{2,2,1} + x_{2,3,1} - x_{2,1,2} - x_{2,1,3} = 1 \\
& x_{2,1,2} + x_{2,3,2} - x_{2,2,1} - x_{2,2,3} = -1 \\
& x_{2,1,3} + x_{2,2,3} - x_{2,3,1} - x_{2,3,2} = 0 \\
& x_{3,2,1} + x_{3,3,1} - x_{3,1,2} - x_{3,1,3} = 0 \\
& x_{3,1,2} + x_{3,3,2} - x_{3,2,1} - x_{3,2,3} = 1 \\
& x_{3,1,3} + x_{3,2,3} - x_{3,3,1} - x_{3,3,2} = -1 \\
& x_{1,1,2} + x_{2,1,2} + x_{3,1,2} \leq 1 \\
& x_{1,2,3} + x_{2,2,3} + x_{3,2,3} \leq 1 \\
& x_{1,3,1} + x_{2,3,1} + x_{3,3,1} \leq 1 \\
& \text{all } x_{q,i,j} \geq 0
\end{aligned}$$

Tractability of Multicommodity Flow Models

Multicommodity flow formulations 10.48 are certainly linear programs. In fact, they are rather special linear programs for which unusually efficient algorithms can be developed. Also, we continue to think in terms of flows and represent complex models in simple diagrams.

Still, very little of the elegant structure presented for single-commodity cases in Sections 10.2 to 10.5 carries over to the multicommodity setting. Most important is a loss of integrality property 10.20.

10.49 Even if all problem data are integer, optimal solutions to multicommodity flow problems may be fractional.

The instance of Sample Exercise 10.29 illustrates. Each commodity is supplied at one node and demanded at another. Since costs on the inner dicycle 1–2–3–1 are zero, all commodities compete for the unit capacities on those arcs. It is easy to verify that the unique optimal solution makes

$$\begin{aligned}
x_{1,1,2} = x_{1,2,3} = x_{1,1,3} &= \tfrac{1}{2} \\
x_{2,2,3} = x_{2,3,1} = x_{2,2,1} &= \tfrac{1}{2} \\
x_{3,3,1} = x_{3,1,2} = x_{3,3,2} &= \tfrac{1}{2}
\end{aligned}$$

Total cost of this fractional solution is $\frac{3}{2}$, versus 2 for any all-integer flow.

origin, not by both origin and destination. Thus, you have as many commodities as there are origins, rather than (number of origins) * (number of destinations). For example, in a 100-city problem, using this observation you would have 100 commodities rather than 10,000 commodities.

Among the biggest examples of multicommodity network problems in existence are the Patient Distribution System models developed by the U.S. Air Force for planning for transport of sick or wounded personnel.

7.8.3 Multicommodity Flow Example

You have decided to compete with Federal Express by offering "point to point" shipment of materials. Starting small, you have identified six cities as the ones you will first serve. In the matrix below is tabulated the average number of tons that potential customers need to move between each origin/destination pair per day. For example, people in city 2 need to move 4 tons per day to city 3.

		Demand in tons, $D(i, j)$, by O/D Pair						Cost/ton Shipped, $C(i, j)$, by Link						Capacity in tons, $U(i, j)$, by Link					
	TO	1	2	3	4	5	6	1	2	3	4	5	6	1	2	3	4	5	6
FROM	1	0	5	9	7	0	4	0	4	5	8	9	9	0	2	3	2	1	20
	2	0	0	4	0	1	0	3	0	3	2	4	6	0	0	2	8	3	9
	3	0	0	0	0	0	0	5	3	0	2	3	5	3	0	0	1	3	9
	4	0	0	0	0	0	0	7	3	3	0	5	6	5	4	6	0	5	9
	5	0	4	0	2	0	8	8	5	3	6	0	3	1	0	2	7	0	9
	6	0	0	0	0	0	0	9	7	4	5	5	0	9	9	9	9	9	0

Rather than use a hub system as Federal Express does, you will ship the materials over a regular directed network. The cost per ton of shipping from any node i to any other node j is denoted by $C(i, j)$. There is an upper limit on the number of tons shipped per day over any link in the network of $U(i, j)$. This capacity restriction applies to the total amount of all goods shipped over that link, regardless of origin or destination. Note that $U(i, j)$ and $C(i, j)$ apply to links in the network, whereas $D(i, j)$ applies to origin/destination pairs. This capacity restriction applies only to the directed flow; e.g., $U(i, j)$ need not equal $U(j, i)$. It may be that none of the goods shipped from origin i to destination j moves over link (i, j). It is important that goods maintain their identity as they move through the network. Notice that city 6 looks like a hub; it has high capacity to and from all other cities.

In order to get a compact formulation, we note that only three cities, 1, 2, and 5 are suppliers. Thus, we need keep track of only three commodities in the network, corresponding to the city of origin of the commodity. Define:

X_{ijk} = tons shipped from city i to city j of commodity k.

The resulting formulation is:


```
MIN     5 X655 + 5 X652 + 5 X651 + 5 X645 + 5 X642 + 5 X641
      + 4 X635 + 4 X632 + 4 X631 + 7 X625 + 7 X622 + 7 X621
      + 9 X615 + 9 X612 + 9 X611 + 3 X565 + 3 X562 + 3 X561
      + 6 X545 + 6 X542 + 6 X541 + 3 X535 + 3 X532 + 3 X531
      + 8 X515 + 8 X512 + 8 X511 + 6 X465 + 6 X462 + 6 X461
      + 5 X455 + 5 X452 + 5 X451 + 3 X435 + 3 X432 + 3 X431
      + 3 X425 + 3 X422 + 3 X421 + 7 X415 + 7 X412 + 7 X411
      + 5 X365 + 5 X362 + 5 X361 + 3 X355 + 3 X352 + 3 X351
      + 2 X345 + 2 X342 + 2 X341 + 5 X315 + 5 X312 + 5 X311
      + 6 X265 + 6 X262 + 6 X261 + 4 X255 + 4 X252 + 4 X251
      + 2 X245 + 2 X242 + 2 X241 + 3 X235 + 3 X232 + 3 X231
      + 9 X165 + 9 X162 + 9 X161 + 9 X155 + 9 X152 + 9 X151
      + 8 X145 + 8 X142 + 8 X141 + 5 X135 + 5 X132 + 5 X131
      + 4 X125 + 4 X122 + 4 X121
SUBJECT TO
! Balance constraints, for each commodity k, city i;
   BAL11)   X611 + X511 + X411 + X311- X161 - X151 - X141
          - X131 - X121 =  -25
   BAL12)   X621 + X421 - X261 - X251 - X241 - X231 + X121 =  5
   BAL13)   X631 + X531 + X431 - X361 - X351 - X341 - X311
          + X231 + X131 =    9
   BAL14)   X641 + X541 - X461 - X451 - X431-X421-X411
          + X341 + X241 + X141 =    7
   BAL15)   X651 - X561 - X541 - X531 - X511 + X451 + X351
          + X251 + X151 =    0
   BAL16) - X651 - X641 - X631 - X621 - X611 + X561 + X461
          + X361 + X261 + X161 =   4
   BAL21)   X612 + X512 + X412 + X312 - X162 - X152 - X142
          - X132 - X122 =    0
   BAL22)   X622 + X422 - X262 - X252 - X242 - X232 + X122 = -5
   BAL23)   X632 + X532 + X432 - X362 - X352 - X342 - X312
          + X232 + X132 =    4
   BAL24)   X642 + X542 - X462 - X452 - X432 - X422 - X412
          + X342 + X242 + X142 =    0
   BAL25)   X652 - X562 - X542 - X532 - X512 + X452 + X352
          + X252 + X152 =    1
   BAL26) - X652 - X642 - X632 - X622 - X612 + X562 + X462
          + X362 + X262 + X162 =    0
   BAL51)   X615 + X515 + X415 + X315 - X165 - X155 - X145
          - X135 - X125 =    0
   BAL52)   X625 + X425 - X265 - X255 - X245 - X235 + X125 =  4
   BAL53)   X635 + X535 + X435 - X365 - X355 - X345 - X315
          + X235 + X135 =    0
   BAL54)   X645 + X545 - X465 - X455 - X435 - X425 - X415
          + X345 + X245 + X145 =    2
   BAL55)   X655 - X565 - X545 - X535 - X515 + X455 + X355
          + X255 + X155 =   -14
   BAL56) - X655 - X645 - X635 - X625 - X615 + X565 + X465
          + X365 + X265 + X165 =    8
! Capacity constraints, for each link i,j;
    CAP12)   X125 + X122 + X121 <=   2
    CAP13)   X135 + X132 + X131 <=   3
    CAP14)   X145 + X142 + X141 <=   2
    CAP15)   X155 + X152 + X151 <=   1
```

```
      CAP16)   X165 + X162 + X161 <=   20
      CAP23)   X235 + X232 + X231 <=   2
      CAP24)   X245 + X242 + X241 <=   8
      CAP25)   X255 + X252 + X251 <=   3
      CAP26)   X265 + X262 + X261 <=   9
      CAP31)   X315 + X312 + X311 <=   3
      CAP34)   X345 + X342 + X341 <=   1
      CAP35)   X355 + X352 + X351 <=   3
      CAP36)   X365 + X362 + X361 <=   9
      CAP41)   X415 + X412 + X411 <=   5
      CAP42)   X425 + X422 + X421 <=   4
      CAP43)   X435 + X432 + X431 <=   6
      CAP45)   X455 + X452 + X451 <=   5
      CAP46)   X465 + X462 + X461 <=   9
      CAP51)   X515 + X512 + X511 <=   1
      CAP53)   X535 + X532 + X531 <=   2
      CAP54)   X545 + X542 + X541 <=   7
      CAP56)   X565 + X562 + X561 <=   9
      CAP61)   X615 + X612 + X611 <=   9
      CAP62)   X625 + X622 + X621 <=   9
      CAP63)   X635 + X632 + X631 <=   9
      CAP64)   X645 + X642 + X641 <=   9
      CAP65)   X655 + X652 + X651 <=   9
END
```

Notice that there are 3 (commodities) × 6 (cities) = 18 balance constraints. If instead we identified goods by origin/destination combination rather than just origin, there would have been 9 × 6 = 54 balance constraints. Solving, we get:

```
        OBJECTIVE FUNCTION VALUE
  1)          361.000000

VARIABLE        VALUE          REDUCED COST
    X641         5.000000          0.000000
    X631         5.000000          0.000000
    X621         3.000000          0.000000
    X565         8.000000          0.000000
    X545         5.000000          0.000000
    X535         1.000000          0.000000
    X531         1.000000          0.000000
    X432         2.000000          0.000000
    X425         4.000000          0.000000
    X345         1.000000          0.000000
    X252         1.000000          0.000000
    X242         2.000000          0.000000
    X232         2.000000          0.000000
    X161        17.000000          0.000000
    X151         1.000000          0.000000
    X141         2.000000          0.000000
    X131         3.000000          0.000000
    X121         2.000000          0.000000

  ROW      SLACK OR SURPLUS      DUAL PRICES
   BAL12)        0.000000        -16.000000
   BAL13)        0.000000        -13.000000
   BAL14)        0.000000        -14.000000
   BAL15)        0.000000         -9.000000
   BAL16)        0.000000         -9.000000
   BAL23)        0.000000         -5.000000
   BAL24)        0.000000         -2.000000
   BAL25)        0.000000         -4.000000
   BAL26)        0.000000         -1.000000
   BAL52)        0.000000         -1.000000
   BAL53)        0.000000          4.000000
   BAL54)        0.000000          2.000000
   BAL55)        0.000000          8.000000
   BAL56)        0.000000          5.000000
   CAP12)        0.000000         12.000000
   CAP13)        0.000000          8.000000
   CAP14)        0.000000          6.000000
   CAP23)        0.000000          2.000000
   CAP53)        0.000000          1.000000
```

Notice that because of capacity limitations on other links, the depot city, 6, is used for many of the shipments.

7.8.4 Fleet Routing and Assignment

An important problem in the airline and trucking industry is fleet routing and assignment. Given a set of shipments or flights to be made, the routing part is concerned with the path that each vehicle takes. The assignment part is of interest if the firm has several different fleets of vehicles available. The question is, then, what type of vehicle is assigned to each flight or shipment. We will describe a simplified version of the approach used by Subramanian et al. (1994) to do fleet assignment at Delta Airlines.

To motivate things, consider the following set of flights serving Chicago (ORD), Denver (DEN) and Los Angeles (LAX) that United Airlines offered at one time on a typical weekday.

Daily Flight Schedule

		City		Time	
	Flight	Depart	Arrive	Depart	Arrive
1	221	ORD	DEN	0800	0934
2	223	ORD	DEN	0900	1039
3	274	LAX	DEN	0800	1116
4	105	ORD	LAX	1100	1314
5	228	DEN	ORD	1100	1423
6	230	DEN	ORD	1200	1521
7	259	ORD	LAX	1400	1609
8	293	DEN	LAX	1400	1510
9	412	LAX	ORD	1400	1959
10	766	LAX	DEN	1600	1912
11	238	DEN	ORD	1800	2121

Negative lengths. We do not generally consider the case of negative lengths, but we want to indicate that negative lengths may cause difficulties, particularly when they lead to cycles of negative length.

10. Find shortest paths in Fig. 461 by inspection and confirm them by Bellman's equations.
11. Find shortest paths in Fig. 462 by inspection and show that Bellman's equations do not have a unique solution.
12. Show by inspection that there are no unique shortest paths in Fig. 463.

Fig. 461. Problem 10

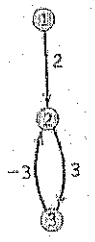


Fig. 462. Problem 11

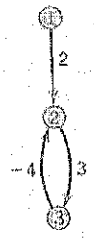


Fig. 463. Problem 12

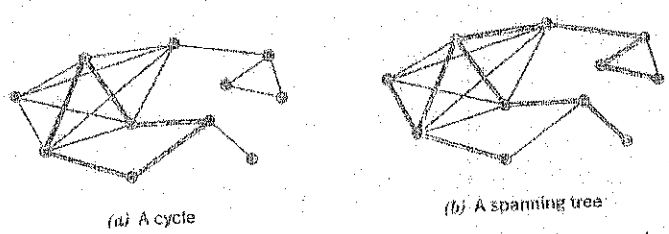
22.4 Shortest Spanning Trees. Kruskal's Greedy Algorithm

So far we have discussed shortest path problems. We now turn to a particularly important kind of graph, called a tree, along with related optimization problems that arise quite often in practice.

By definition, a **tree** T is a graph that is connected and has no cycles. "Connected" means that there is a path from any vertex in T to any other vertex in T. A **cycle**[7] is a path $s \to t$ of at least three edges that is closed ($t = s$); see also Sec. 22.2. Figure 464a shows an example.

A **spanning tree** T in a given connected graph $G = (V, E)$ is a tree containing *all* the n vertices of G. See Fig. 464b. Such a tree has $n - 1$ edges. (Proof?)

A **shortest spanning tree** T in a connected graph G (whose edges (i, j) have lengths $l_{ij} > 0$) is a spanning tree for which Σl_{ij} (sum over all edges of T) is minimum compared to Σl_{ij} for any other spanning tree in G.



(a) A cycle (b) A spanning tree

Fig. 464. Example of (a) a cycle, (b) a spanning tree in a graph

[7] Or **circuit**. Caution! The terminology varies considerably.

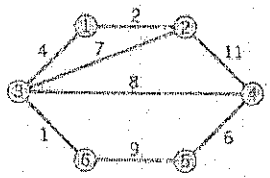


Fig. 465. Graph in Example 1

EXAMPLE 1 **Application of Kruskal's algorithm**
Find a shortest spanning tree in the graph in Fig. 465.

Solution. See Table 22.4. In some of the intermediate stages the edges chosen form a *disconnected* graph (see Fig. 466); this is typical. We stop after $n - 1 = 5$ choices since a spanning tree has $n - 1$ edges. In our problem the edges chosen are in the upper part of the list. This is typical of problems of any size; in general, edges farther down in the list have a smaller chance of being chosen.

Table 22.4
Solution in Example 1

Edge	Length	Choice
(3, 6)	1	1st
(1, 2)	2	2nd
(1, 3)	4	3rd
(4, 5)	6	4th
(2, 3)	7	Reject
(3, 4)	8	5th
(5, 6)	9	
(2, 4)	11	

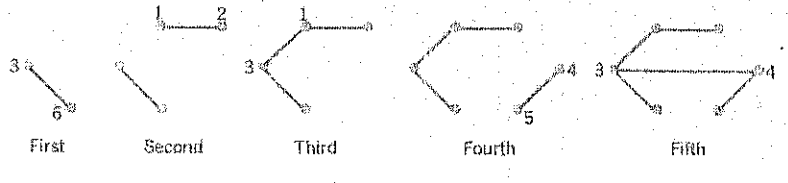


Fig. 466. Choice process in Example 1

The efficiency of Kruskal's method is greatly increased by

Double labeling of vertices
Each vertex i carries a double label (r_i, p_i), where
r_i = *Root of the subtree to which i belongs,*
p_i = *Predecessor of i in its subtree,*
$p_i = 0$ *for roots.*

This simplifies

Rejecting. *If (i, j) is next in the list to be considered, reject (i, j) if $r_i = r_j$* (that is, i and j are in the same subtree, so that they are already joined by edges and (i, j) would thus create a cycle). *If $r_i \neq r_j$, include (i, j) in T.*

Prim's algorithm shown in Table 22.6 is another popular algorithm for the shortest spanning tree problem (see Sec. 22.4). This algorithm gives a tree T at each stage, a property that Kruskal's algorithm in the last section did not have (look back at Fig. 466 if you did not notice it).

In Prim's algorithm, starting from any single vertex, which we call 1, we "grow" the tree T by adding edges to it, one at a time, according to some rule (below) until T finally becomes a *spanning* tree, which is shortest.

We denote by U the set of vertices of the growing tree T and by S the set of its edges. Thus, initially $U = \{1\}$ and $S = \emptyset$; at the end, $U = V$, the vertex set of the given graph $G = (V, E)$, whose edges (i, j) have length $l_{ij} > 0$, as before.

Thus at the beginning (Step 1) the labels $\lambda_2, \cdots, \lambda_n$ of the vertices $2, \cdots, n$ are the lengths of the edges connecting them to vertex 1 (or ∞ if there is no such edge in G). And we pick (Step 2) the shortest of these as the first edge of the growing tree T and include its other end j in U (choosing the smallest j if there are several, to make the process unique). Updating labels in Step 3 (at this stage and at any later stage) concerns each vertex k not yet in U. Vertex k has label $\lambda_k = l_{i(k),k}$ from before. If $l_{jk} < \lambda_k$, this means that k is closer to the new member j just included in U than k is to its old "closest neighbor" $i(k)$ in U. Then we update the label of k, replacing $\lambda_k = l_{i(k),k}$ by $\lambda_k = l_{jk}$ and setting $i(k) = j$. If, however, $l_{jk} \geqq \lambda_k$ (the *old* label of k), we don't touch the old label. Thus the label λ_k always identifies the closest neighbor of k in U, and this is updated in Step 3 as U and the tree T grow. From the final labels we can backtrack the final tree, and from their numerical values we compute the total length (sum of the lengths of the edges) of this tree.

EXAMPLE 1 **Application of Prim's algorithm**

Find a shortest spanning tree in the graph in Fig. 468 (which is the same as in Example 1, Sec. 22.4, so that we can compare).

Solution. The steps are as follows.

1. $i(k) = 1$, $U = \{1\}$, $S = \emptyset$, initial labels see Table 22.7.
2. $\lambda_2 = l_{12} = 2$ is smallest, $U = \{1, 2\}$, $S = \{(1, 2)\}$
3. Update labels as shown in Table 22.7, column (I).
2. $\lambda_3 = l_{13} = 4$ is smallest, $U = \{1, 2, 3\}$, $S = \{(1, 2), (1, 3)\}$
3. Update labels as shown in Table 22.7, column (II).
2. $\lambda_6 = l_{36} = 1$ is smallest, $U = \{1, 2, 3, 6\}$, $S = \{(1, 2), (1, 3), (3, 6)\}$
3. Update labels as shown in Table 22.7, column (III).
2. $\lambda_4 = l_{34} = 8$ is smallest, $U = \{1, 2, 3, 4, 6\}$, $S = \{(1, 2), (1, 3), (3, 4), (3, 6)\}$
3. Update labels as shown in Table 22.7, column (IV).
2. $\lambda_5 = l_{45} = 6$ is smallest, $U = V$, $S = (1, 2), (1, 3) (3, 4), (3, 6), (4, 5)$. Stop.

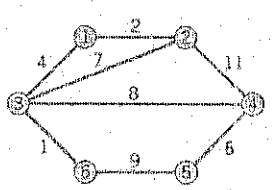


Fig. 468. Graph in Example 1

Trees are among the most important types of graphs, and they occur in various applications. Familiar examples are family trees and organization charts. Trees can be used to exhibit, organize, or analyze electrical networks, producer–consumer and other business relations, information in database systems, syntactic structure of computer programs, etc. We mention a few specific applications that need no lengthy additional explanations.

The set of shortest paths from vertex 1 to the vertices $2, \cdots, n$ in the last section forms a spanning tree.

Railway lines connecting a number of cities (the vertices) can be set up in the form of a spanning tree, the "length" of a line (edge) being the construction cost, and one wants to minimize the total construction cost. Similarly for bus lines, where "length" may be the average annual operating cost. Or for steamship lines (freight lines), where "length" may be profit and the goal is the maximization of total profit. Or in a network of telephone lines between some cities, a shortest spanning tree may simply represent a selection of lines that connect all the cities at minimal cost.

As a somewhat more sophisticated example, consider a private communication network G, let p_{ij} be the probability of intercepting line (i, j) by an outsider, and suppose that one wants to communicate a confidential message to all participants (vertices) along a spanning tree T in G that minimizes the product of the p_{ij} of all the edges (lines) of T, that is, assuming stochastic independence, the total probability of interception; equivalently, a tree T that minimizes the sum of the logarithms of the p_{ij} or, better, the sum of $l_{ij} = \ln \tilde{p}_{ij}$, where $\tilde{p}_{ij} = Kp_{ij}$ with K so large that $\tilde{p}_{ij} > 1$ for each edge of the network G, thus $l_{ij} > 0$.

In addition to these examples from transportation and communication networks, one could mention others from distribution networks, and so on.

We shall now discuss a simple algorithm for the shortest spanning tree problem, which is particularly suitable for sparse graphs (graphs with very few edges; see Sec. 22.1). This algorithm is shown in Table 22.3.

Table 22.3
Kruskal's Greedy Algorithm for Shortest Spanning Trees[a]

ALGORITHM KRUSKAL [$G = (V, E)$, l_{ij} for all (i, j) in E]
Given a connected graph $G = (V, E)$ with edges (i, j) having length $l_{ij} > 0$, the algorithm determines a shortest spanning tree T in G.

INPUT: Edges (i, j) of G and their lengths l_{ij}
OUTPUT: Shortest spanning tree T in G

1. Order the edges of G in ascending order of length.
2. Choose them in this order as edges of T, rejecting an edge only if it forms a cycle with edges already chosen.
   If $n - 1$ edges have been chosen, then
   OUTPUT T (= the set of edges chosen). Stop

End KRUSKAL

[a] *Proceedings of the American Mathematical Society* 7 (1956), 48–50.

Negative lengths. We do not generally consider the case of negative lengths, but we want to indicate that negative lengths may cause difficulties, particularly when they lead to cycles of negative length.

10. Find shortest paths in Fig. 461 by inspection and confirm them by Bellman's equations.
11. Find shortest paths in Fig. 462 by inspection and show that Bellman's equations do not have a unique solution.
12. Show by inspection that there are no unique shortest paths in Fig. 463.

Fig. 461. Problem 10

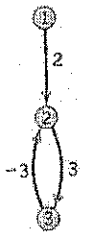


Fig. 462. Problem 11

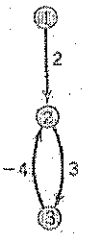


Fig. 463. Problem 12

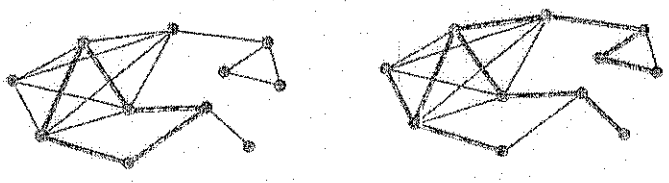
22.4 Shortest Spanning Trees. Kruskal's Greedy Algorithm

So far we have discussed shortest path problems. We now turn to a particularly important kind of graph, called a tree, along with related optimization problems that arise quite often in practice.

By definition, a **tree** T is a graph that is connected and has no cycles. "**Connected**" means that there is a path from any vertex in T to any other vertex in T. A **cycle**[7] is a path $s \to t$ of at least three edges that is closed ($t = s$); see also Sec. 22.2. Figure 464a shows an example.

A **spanning tree** T in a given connected graph $G = (V, E)$ is a tree containing *all* the n vertices of G. See Fig. 464b. Such a tree has $n - 1$ edges. (Proof?)

A **shortest spanning tree** T in a connected graph G (whose edges (i, j) have lengths $l_{ij} > 0$) is a spanning tree for which Σl_{ij} (sum over all edges of T) is minimum compared to Σl_{ij} for any other spanning tree in G.



(a) A cycle *(b)* A spanning tree

Fig. 464. Example of (a) a cycle, (b) a spanning tree in a graph

[7]Or **circuit**. Caution! The terminology varies considerably.

Trees are among the most important types of graphs, and they occur in various applications. Familiar examples are family trees and organization charts. Trees can be used to exhibit, organize, or analyze electrical networks, producer–consumer and other business relations, information in database systems, syntactic structure of computer programs, etc. We mention a few specific applications that need no lengthy additional explanations.

The set of shortest paths from vertex 1 to the vertices $2, \cdots, n$ in the last section forms a spanning tree.

Railway lines connecting a number of cities (the vertices) can be set up in the form of a spanning tree, the "length" of a line (edge) being the construction cost, and one wants to minimize the total construction cost. Similarly for bus lines, where "length" may be the average annual operating cost. Or for steamship lines (freight lines), where "length" may be profit and the goal is the maximization of total profit. Or in a network of telephone lines between some cities, a shortest spanning tree may simply represent a selection of lines that connect all the cities at minimal cost.

As a somewhat more sophisticated example, consider a private communication network G, let p_{ij} be the probability of intercepting line (i, j) by an outsider, and suppose that one wants to communicate a confidential message to all participants (vertices) along a spanning tree T in G that minimizes the product of the p_{ij} of all the edges (lines) of T, that is, assuming stochastic independence, the total probability of interception; equivalently, a tree T that minimizes the sum of the logarithms of the p_{ij} or, better, the sum of $l_{ij} = \ln \bar{p}_{ij}$, where $\bar{p}_{ij} = Kp_{ij}$ with K so large that $\bar{p}_{ij} > 1$ for each edge of the network G, thus $l_{ij} > 0$.

In addition to these examples from transportation and communication networks, one could mention others from distribution networks, and so on.

We shall now discuss a simple algorithm for the shortest spanning tree problem, which is particularly suitable for sparse graphs (graphs with very few edges; see Sec. 22.1). This algorithm is shown in Table 22.3.

Table 22.3
Kruskal's Greedy Algorithm for Shortest Spanning Trees[6]

ALGORITHM KRUSKAL [$G = (V, E)$, l_{ij} for all (i, j) in E]
Given a connected graph $G = (V, E)$ with edges (i, j) having length $l_{ij} > 0$, the algorithm determines a shortest spanning tree T in G.

INPUT: Edges (i, j) of G and their lengths l_{ij}
OUTPUT: Shortest spanning tree T in G

1. Order the edges of G in ascending order of length.
2. Choose them in this order as edges of T, rejecting an edge only if it forms a cycle with edges already chosen.

If $n - 1$ edges have been chosen, then
OUTPUT T (= the set of edges chosen). Stop
End KRUSKAL

[6] *Proceedings of the American Mathematical Society* 7 (1956), 48–50.

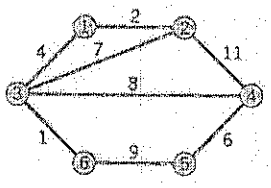


Fig. 465. Graph in Example 1

EXAMPLE 1 **Application of Kruskal's algorithm**

Find a shortest spanning tree in the graph in Fig. 465.

Solution. See Table 22.4. In some of the intermediate stages the edges chosen form a *disconnected* graph (see Fig. 466); this is typical. We stop after $n - 1 = 5$ choices since a spanning tree has $n - 1$ edges. In our problem the edges chosen are in the upper part of the list. This is typical of problems of any size; in general, edges farther down in the list have a smaller chance of being chosen.

Table 22.4
Solution in Example 1

Edge	Length	Choice
(3, 6)	1	1st
(1, 2)	2	2nd
(1, 3)	4	3rd
(4, 5)	6	4th
(2, 3)	7	Reject
(3, 4)	8	5th
(5, 6)	9	
(2, 4)	11	

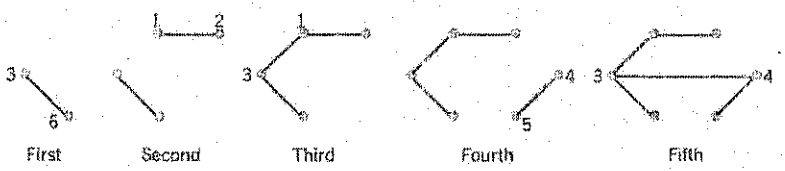


Fig. 466. Choice process in Example 1

The efficiency of Kruskal's method is greatly increased by

Double labeling of vertices

Each vertex i carries a double label (r_i, p_i), where

r_i = *Root of the subtree to which i belongs,*

p_i = *Predecessor of i in its subtree,*

$p_i = 0$ *for roots.*

This simplifies

Rejecting. *If (i, j) is next in the list to be considered, reject (i, j) if $r_i = r_j$* (that is, i and j are in the same subtree, so that they are already joined by edges and (i, j) would thus create a cycle). *If $r_i \neq r_j$, include (i, j) in T.*

Prim's algorithm shown in Table 22.6 is another popular algorithm for the shortest spanning tree problem (see Sec. 22.4). This algorithm gives a tree T at each stage, a property that Kruskal's algorithm in the last section did not have (look back at Fig. 466 if you did not notice it).

In Prim's algorithm, starting from any single vertex, which we call 1, we "grow" the tree T by adding edges to it, one at a time, according to some rule (below) until T finally becomes a *spanning* tree, which is shortest.

We denote by U the set of vertices of the growing tree T and by S the set of its edges. Thus, initially $U = \{1\}$ and $S = \emptyset$; at the end, $U = V$, the vertex set of the given graph $G = (V, E)$, whose edges (i, j) have length $l_{ij} > 0$, as before.

Thus at the beginning (Step 1) the labels $\lambda_2, \cdots, \lambda_n$ of the vertices $2, \cdots, n$ are the lengths of the edges connecting them to vertex 1 (or ∞ if there is no such edge in G). And we pick (Step 2) the shortest of these as the first edge of the growing tree T and include its other end j in U (choosing the smallest j if there are several, to make the process unique). Updating labels in Step 3 (at this stage and at any later stage) concerns each vertex k not yet in U. Vertex k has label $\lambda_k = l_{i(k),k}$ from before. If $l_{jk} < \lambda_k$, this means that k is closer to the new member j just included in U than k is to its old "closest neighbor" $i(k)$ in U. Then we update the label of k, replacing $\lambda_k = l_{i(k),k}$ by $\lambda_k = l_{jk}$ and setting $i(k) = j$. If, however, $l_{jk} \geqq \lambda_k$ (the *old* label of k), we don't touch the old label. Thus the label λ_k always identifies the closest neighbor of k in U, and this is updated in Step 3 as U and the tree T grow. From the final labels we can backtrack the final tree, and from their numerical values we compute the total length (sum of the lengths of the edges) of this tree.

EXAMPLE 1 Application of Prim's algorithm

Find a shortest spanning tree in the graph in Fig. 468 (which is the same as in Example 1, Sec. 22.4, so that we can compare).

Solution. The steps are as follows.

1. $i(k) = 1$, $U = \{1\}$, $S = \emptyset$, initial labels see Table 22.7.
2. $\lambda_2 = l_{12} = 2$ is smallest, $U = \{1, 2\}$, $S = \{(1, 2)\}$
3. Update labels as shown in Table 22.7, column (I).
2. $\lambda_3 = l_{13} = 4$ is smallest, $U = \{1, 2, 3\}$, $S = \{(1, 2), (1, 3)\}$
3. Update labels as shown in Table 22.7, column (II).
2. $\lambda_6 = l_{36} = 1$ is smallest, $U = \{1, 2, 3, 6\}$, $S = \{(1, 2), (1, 3), (3, 6)\}$
3. Update labels as shown in Table 22.7, column (III).
2. $\lambda_4 = l_{34} = 8$ is smallest, $U = \{1, 2, 3, 4, 6\}$, $S = \{(1, 2), (1, 3), (3, 4), (3, 6)\}$
3. Update labels as shown in Table 22.7, column (IV).
2. $\lambda_5 = l_{45} = 6$ is smallest, $U = V$, $S = (1, 2), (1, 3) (3, 4), (3, 6), (4, 5)$. Stop.

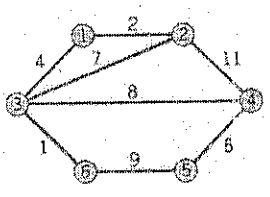


Fig. 468. Graph in Example 1

指派問題

$$\text{Min } Z=\sum_{i=1}^{n}\sum_{j=1}^{n} C_{ij}X_{ij} \quad \left(\text{Max } Z=\sum_{i=1}^{n}\sum_{j=1}^{n} C_{ij}X_{ij}\right)$$

成本 / 效率

$$\text{s.t. } \sum_{i=1}^{n} X_{ij}=1 \quad (j=1,\dots,n)$$

$$\sum_{j=1}^{n} X_{ij}=1 \quad (i=1,\dots,n)$$

多以 K. W. Kuhn 提出的匈牙利法 (Hungarian Method) 求解.

	趙	錢	孫	李	王
仰式	37	32	(33)	37	35
蝶式	43	33	42	(34)	41
蛙式	33	(28)	38	30	33
自由式	(29)	26	29	28	31

Total Cost = 124

```
min 37x11+32x12+33x13+37x14+35x15+
    43x21+33x22+42x23+34x24+41x25+
    33x31+28x32+38x33+30x34+33x35+
    29x41+26x42+29x43+28x44+31x45
st
    x11+ x12+ x13+ x14+ x15=1
    x21 +x22+ x23+ x24+ x25=1
    x31 +x32+ x33+ x34+ x35=1
    x41 +x42+ x43+ x44+ x45=1
    x11+ x21+ x31+ x41 <=1
    x12+ x22+ x32+ x42 <=1
    x13+ x23+ x33+ x43 <=1
    x14+ x24+ x34+ x44 <=1
    x15+ x25+ x35+ x45 <=1
end
inte 20
```

指派問題

$$\text{Min } Z=\sum_{i=1}^{n}\sum_{j=1}^{n} C_{ij}X_{ij} \quad \left(\text{Max } Z=\sum_{i=1}^{n}\sum_{j=1}^{n} C_{ij}X_{ij}\right)$$

（C_{ij}：成本　效率）

$$\text{s.t. } \sum_{i=1}^{n} X_{ij}=1 \quad (j=1,\dots,n)$$

$$\sum_{j=1}^{n} X_{ij}=1 \quad (i=1,\dots,n)$$

多以 K. W. Kuhn 提出的匈牙利法（Hungarian Method）求解：

	趙	錢	孫	李	王
仰式	37	32	(33)	37	35
蝶式	43	33	42	(34)	41
蛙式	33	(28)	38	30	33
自由式	(29)	26	29	28	31

Total Cost = 124

```
min 37x11+32x12+33x13+37x14+35x15+
      43x21+33x22+42x23+34x24+41x25+
      33x31+28x32+38x33+30x34+33x35+
      29x41+26x42+29x43+28x44+31x45
st
     x11+ x12+ x13+ x14+ x15=1
     x21 +x22+ x23+ x24+ x25=1
     x31 +x32+ x33+ x34+ x35=1
     x41 +x42+ x43+ x44+ x45=1
     x11+ x21+ x31+ x41 <=1
     x12+ x22+ x32+ x42 <=1
     x13+ x23+ x33+ x43 <=1
     x14+ x24+ x34+ x44 <=1
     x15+ x25+ x35+ x45 <=1
end
inte 20
```

subject to

$$2x_1 + 2x_2 + x_3 = 8$$
$$4x_1 + 2x_2 + x_4 = 6 - 2t$$
$$x_1, x_2, x_3, x_4 \geq 0$$

where $-\infty \leq t \leq \infty$. The parametric analysis of the problem yields the following results:

$-\infty \leq t \leq -5$: Optimal basis is $\mathbf{B} = (\mathbf{P}_1, \mathbf{P}_4)$

$-5 \leq t \leq -1$:Optimal basis is $\mathbf{B} = (\mathbf{P}_1, \mathbf{P}_2)$

$-1 \leq t \leq 2$:Optimal basis is $\mathbf{B} = (\mathbf{P}_2, \mathbf{P}_3)$

Determine all the critical values of t that may exist for $t \geq 2$.

CHAPTER 8

Goal Programming

The LP models presented in the preceding chapters are based on the optimization of a *single* objective function. There are situations where multiple (possibly conflicting) objectives may be more appropriate. For example, aspiring politicians may promise to reduce the national debt and, simultaneously, offer income tax relief. In such situations, it may be impossible to find a single solution that optimizes the conflicting objectives. Instead, we may seek a *compromise* solution based on the relative importance of each objective.

This chapter presents the goal programming technique for solving multiobjective models. The principal idea is to convert the original multiple objectives into a single goal. The resulting model yields what is usually referred to as an **efficient solution** because it may not be optimum with respect to *all* the conflicting objectives of the problem.

8.1 A GOAL PROGRAMMING FORMULATION

The idea of goal programming is illustrated by an example.

Example 8.1-1

Fairville is a small city with a population of about 20,000 residents. The city council is in the process of developing an equitable city tax rate table. The annual taxation base for real estate property is $550 million. The annual taxation bases for food and drugs and for general sales are $35 million and $55 million, respectively. Annual local gasoline consumption is estimated at 7.5 million gallons. The city council wants to develop the tax rates based on four main goals.

1. Tax revenues must be at least $16 million to meet the city's financial commitments.
2. Food and drug taxes cannot exceed 10% of all taxes collected.

3. General sales taxes cannot exceed 20% of all taxes collected.
4. Gasoline tax cannot exceed 2 cents per gallon.

Let the variables x_p, x_f, and x_s represent the tax rates (expressed as proportions of taxation bases) for property, food and drugs, and general sales; and define the variable x_g as the gasoline tax in cents per gallon. The goals of the city council are then expressed as

$$\begin{aligned} 550x_p + 35x_f + 55x_s + .075x_g &\geq 16 && \text{(Tax revenue)}\\ 35x_f &\leq .1(550x_p + 35x_f + 55x_s + .075x_g) && \text{(Food/drug tax)}\\ 55x_s &\leq .2(550x_p + 35x_f + 55x_s + .075x_g) && \text{(General tax)}\\ x_g &\leq 2 && \text{(Gasoline tax)}\\ x_p, x_f, x_s, x_g &\geq 0 \end{aligned}$$

These constraints are then simplified as

$$\begin{aligned} 550x_p + 35x_f + 55x_s + .075x_g &\geq 16\\ 55x_p - 31.5x_f + 5.5x_s + .0075x_g &\geq 0\\ 110x_p + 7x_f - 44x_s + .015x_g &\geq 0\\ x_g &\leq 2\\ x_p, x_f, x_s, x_g &\geq 0 \end{aligned}$$

Each of the inequalities of the model represents a goal that the city council aspires to satisfy. Most likely, however, the best we can do is seek a compromise solution among these conflicting goals.

The manner in which goal programming finds a compromise solution is to convert each inequality into a flexible goal in which the corresponding constraint may be violated, if necessary. In terms of the Fairville model, the flexible goals are expressed as follows:

$$\begin{aligned} 550x_p + 35x_f + 55x_s + .075x_g + s_1^+ - s_1^- &= 16\\ 55x_p - 31.5x_f + 5.5x_s + .0075x_g + s_2^+ - s_2^- &= 0\\ 110x_p + 7x_f - 44x_s + .015x_g + s_3^+ - s_3^- &= 0\\ x_g + s_4^+ - s_4^- &= 2\\ x_p, x_f, x_s, x_g &\geq 0\\ s_i^+, s_i^- &\geq 0,\ i = 1, 2, 3, 4 \end{aligned}$$

The nonnegative variables s_i^+ and s_i^-, $i = 1, 2, 3, 4$, are called **deviational variables** because they represent the deviations above and below the right-hand side of constraint i.

The deviational variables s_i^+ and s_i^- are by definition dependent and, hence, cannot be basic variables simultaneously. This means that in any simplex iteration, at most *one* of the two deviational variables can assume a positive value. If the original ith inequality is of the type $\leq$ and its $s_i^+ > 0$, then the ith goal will be satisfied; otherwise, if $s_i^- > 0$, goal i will not be satisfied. In essence, the definition of s_i^+ and s_i^- allows us to meet or violate the ith goal at will. This is the type of flexibility that characterizes goal programming when it seeks a compromise solution. Naturally, a good compromise solution aims at minimizing the amount by which each goal is violated.

In the Fairville model, given that the first three constraints are of the type $\geq$ and the fourth constraint is of the type $\leq$, the deviational variables s_1^+, s_2^+, s_3^+, and s_4^- of the problem represent the amounts by which the respective goals are violated. Thus, the compromise solution tries to satisfy the following four objectives as much as possible:

$$\begin{aligned} \text{Minimize } G_1 &= s_1^+\\ \text{Minimize } G_2 &= s_2^+\\ \text{Minimize } G_3 &= s_3^+\\ \text{Minimize } G_4 &= s_4^- \end{aligned}$$

These functions are minimized subject to the constraint equations of the model.

How can we optimize a multiobjective model with possibly conflicting goals? Two methods have been developed for this purpose: (1) the weights method and (2) the preemptive method. Both methods are based on converting the multiple objectives into a single function as detailed in Section 8.2.

PROBLEM SET 8.1A

1. Formulate the Fairville tax problem, assuming that the town council is specifying an additional goal, G_5, that requires gasoline tax to equal at least 10% of the total tax bill.
2. The NW Shopping Mall conducts special events to attract potential patrons. The two most popular events that seem to attract teenagers, the young/middle-aged group, and senior citizens are band concerts and art and craft shows. The costs per presentation of the band and art show are \$1500 and \$3000, respectively. The total (strict) annual budget allocated to the two events is \$15,000. The mall manager estimates the attendance of the events as follows:

	Number attending per presentation		
Event	*Teenagers*	*Young/middle age*	*Seniors*
Band concert	200	100	0
Art show	0	400	250

The manager has set the minimum annual goals of 1000, 1200, and 800 for the attendance of teenagers, the young/middle-aged group, and seniors, respectively. Formulate the problem as a goal programming model.

3. Ozark University admissions office is processing freshman applications for the upcoming academic year. The applications fall into three categories: instate, out-of-state, and international. The male–female ratios for in-state and out-of-state applicants are 1:1 and 3:2, respectively. For the international students, the corresponding ratio is 8:1. The American College Test (ACT) score is an important factor in accepting new students.

11. *Chebyshev Problem.* An alternative goal for the regression model in Problem 10 is to minimize over b_j the maximum of the absolute deviations. Formulate the problem as a goal programming model.

8.2 GOAL PROGRAMMING ALGORITHMS

This section presents two algorithms for solving goal programming. Both methods convert the multiple goals into a single objective function. In the **weights method**, the single objective function is the weighted sum of the functions representing the goals of the problem. The **preemptive method** starts by prioritizing the goals in order of importance. The model is then optimized using one goal at a time such that the optimum value of a higher priority goal is never degraded by a lower priority goal.

The proposed two methods do not generally produce the same solution. Neither method, however, is superior to the other because each technique is designed to satisfy certain decision-making preferences.

8.2.1 The Weights Method

Suppose that the goal programming model has n goals and that the ith goal is given as

$$\text{Minimize } G_i,\ i = 1, 2, \ldots, n$$

The combined objective function used in the weights method is defined as

$$\text{Minimize } z = w_1G_1 + w_2G_2 + \ldots + w_nG_n$$

The parameter w_i, $i = 1, 2, \ldots, n$, represents positive weights that reflect the decision maker's preferences regarding the relative importance of each goal. For example, $w_i = 1$, for all i, signifies that all goals carry equal weights. The determination of the specific values of these weights is subjective. Indeed, the apparently sophisticated analytic procedures developed in the literature (see, e.g., Cohon, 1978) are still rooted in subjective assessments.

Example 8.2-1

TopAd, a new advertising agency with 10 employees, has received a contract to promote a new product. The agency can advertise by radio and television. The following table provides data about the number of people reached by each type of advertisement, and the cost and labor requirements.

	Data/min advertisement	
	Radio	*Television*
Exposure (in millions of persons)	4	8
Cost (in thousands of dollars)	8	24
Assigned employees	1	2

The contract prohibits TopAd from using more than 6 minutes of radio advertisement. Additionally, radio and television advertisements need to reach at least 45 million people. TopAd has set a budget goal of $100,000 for the project. How many minutes of radio and television advertisement should TopAd use?

Let x_1 and x_2 be the minutes allocated to radio and television advertisements. The goal programming formulation for the problem is given as

$$\text{Minimize } G_1 = s_1^+ \text{ (Satisfy exposure goal)}$$

$$\text{Minimize } G_2 = s_2^- \text{ (Satisfy budget goal)}$$

subject to

$$\begin{aligned}
4x_1 + 8x_2 + s_1^+ - s_1^- &= 45 && \text{(Exposure goal)}\\
8x_1 + 24x_2 + s_2^+ - s_2^- &= 100 && \text{(Budget goal)}\\
x_1 + 2x_2 &\le 10 && \text{(Personnel limit)}\\
x_1 &\le 6 && \text{(Radio limit)}
\end{aligned}$$

$$x_1, x_2, s_1^+, s_1^-, s_2^+, s_2^- \ge 0$$

TopAd's management assumes that the exposure goal is twice as important as the budget goal. The combined objective function thus becomes

$$\text{Minimize } z = 2G_1 + G_2 = 2s_1^+ + s_2^-$$

The optimum solution (obtained by TORA) is

$$z = 10$$

$$x_1 = 5 \text{ minutes},\ x_2 = 2.5 \text{ minutes},\ s_1^+ = 5 \text{ million persons}$$

All the remaining variables equal zero.

The fact that the optimum value of z is not zero indicates that at least one of the goals is not met. Specifically, $s_1^+ = 5$ means that the exposure goal (of at least 45 million persons) is missed by 5 million individuals. Conversely, the budget goal (of not exceeding $100,000) is not violated because $s_2^- = 0$.

Goal programming yields only an *efficient* solution to the problem, which is not necessarily optimum. For example, the solution $x_1 = 6$ and $x_2 = 2$ yields the same exposure ($4 \times 6 + 8 \times 2 = 40$ million persons) but costs less ($8 \times 6 + 24 \times 2 = \$96{,}000$). In essence, what goal programming does is to find a solution that simply *satisfies* the goals of the model with no regard to optimization. Such "deficiency" in finding an optimum solution raises doubts about the viability of goal programming as an optimizing technique (see Example 8.2-3 for further discussion).

PROBLEM SET 8.2A

1. Consider Problem 1, Set 8.1a dealing with the Fairville tax situation. Solve the problem, assuming that all five goals have the same weight. Does the solution satisfy all the goals?
2. In Problem 2, Set 8.1a, suppose that the goal of attracting young/middle-aged people is twice as important as either of the other two categories (teens and seniors). Find the associated solution, and check if all the goals have been met.
3. In the Ozark University admission situation described in Problem 3, Set 8.1a, suppose that the limit on the size of the incoming freshman class must be met, but the remaining

requirements can be treated as flexible goals. Further, assume that the ACT score goal is twice as important as any of the remaining goals.

(a) Solve the problem, and specify whether or not all the goals are satisfied.

(b) If, in addition, the size of the incoming class can be treated as a flexible goal that is twice as important as the ACT goal, how would this change affect the solution?

4. In the Circle K model of Problem 4, Set 8.1a, is it possible to satisfy all the nutritional requirements?

5. In Problem 5, Set 8.1a, determine the solution, and specify whether or not the daily production of wheels and seats can be balanced.

6. In Problem 6, Set 8.1a, suppose that the market demand goal is twice as important as that of balancing the two machines, and that no overtime is allowed. Solve the problem, and determine if the goals are met.

7. In Problem 7, Set 8.1a, suppose that the production quota for the two products needs to be met, using overtime if necessary. Find a solution to the problem, and specify the amount of overtime, if any, needed to meet the production quota.

8. In the Vista City Hospital of Problem 8, Set 8.1a, suppose that only the bed limits represent flexible goals and that all the goals have equal weights. Can all the goals be met?

9. The Malco Company has compiled the following table from the files of five of its employees to study the relationship between income and age, education (expressed in number of college years completed), and experience (expressed in number of years in the business).

Age (yr)	Education (yr)	Experience (yr)	Annual income ($)
30	4	5	40,000
39	5	10	48,000
44	2	14	38,000
48	0	18	36,000
37	3	9	41,000

Use the goal programming formulation in Problem 10, Set 8.1a to fit the data into the linear equation $y = b_0 + b_1x_1 + b_2x_2 + b_3x_3$.

10. Solve Problem 9 using the Chebyshev Method proposed in Problem 11, Set 8.1a.

8.2.2 The Preemptive Method

In the preemptive method, the decision maker must rank the goals of the problem in order of importance. Given an n-goal situation, the objectives of the problem are written as

$$\text{Minimize } G_1 = \rho_1 \text{ (Highest priority)}$$

$$\vdots$$

$$\text{Minimize } G_n = \rho_n \text{ (Lowest priority)}$$

The variable ρ_i is either s_i^+ or s_i^- representing goal i. For example, in the TopAd model (Example 8.2-1), $\rho_1 = s_1^+$ and $\rho_2 = s_2^-$.

The solution procedure considers one goal at a time, starting with the highest priority, G_1, and terminating with the lowest, G_n. *The process is carried out such that the solution obtained from a lower priority goal never degrades any higher priority solutions.*

The literature on goal programming presents a "special" simplex method that guarantees the nondegradation of higher priority solutions. The method uses the **column-dropping rule** that calls for eliminating a *nonbasic* variable x_j with $z_j - c_j \neq 0$ from the optimal tableau of goal G_k before solving the problem of goal G_{k+1}. The rule recognizes that such nonbasic variables, if elevated above zero level in the optimization of succeeding goals, can degrade (but never improve) the quality of a higher priority goal. The procedure requires modifying the simplex tableau so that it will carry the objective functions of all the goals of the model.

The proposed column-dropping modification needlessly complicates goal programming. In this presentation, we show that the same results can be achieved in a more straightforward manner using the following steps:

Step 0. Identify the goals of the model and rank them in order of priority:

$$G_1 = \rho_1 \succ G_2 = \rho_2 \succ \ldots \succ G_n = \rho_n$$

Set $i = 1$.

Step i. Solve LP_i that minimizes G_i, and let $\rho_i = \rho_i^*$ define the corresponding optimum value of the deviational variable ρ_i. If $i = n$, stop; LP_n solves the n-goal program. Otherwise, augment the constraint $\rho_i = \rho_i^*$ to the constraints of the G_i-problem to ensure that the value of ρ_i will not be degraded in future problems. Set $i = i + 1$, and repeat step i.

The successive addition of the special constraints $\rho_i = \rho_i^*$ may not be as "elegant" theoretically as the *column-dropping rule.* Nevertheless, it achieves the exact same result. More important, it is easier to understand.

Some may argue that the column-dropping rule offers computational advantages. Essentially, the rule makes the problem smaller successively by removing variables, whereas our procedure makes the problem larger by adding new constraints. However, considering the nature of the additional constraints ($\rho_i = \rho_i^*$), we should be able to modify the simplex algorithm to implement the additional constraint implicitly through direct substitution of the variable ρ_i. This substitution affects only the constraint in which ρ_i appears and, in effect, reduces the number of variables as we move from one goal to the next. Alternatively, we can use the bounded simplex method of Section 7.3 by replacing $\rho_i = \rho_i^*$ with $\rho_i \leq \rho_i^*$, in which case the additional constraints are accounted for implicitly. In this regard, the column-dropping rule, theoretical appeal aside, does not appear to offer a particular computational advantage. For the sake of completeness, however, we will demonstrate in Example 8.2-3 how the column-dropping rule works.

Example 8.2-2

The problem of Example 8.2-1 is solved by the preemptive method. Assume that the exposure goal has a higher priority.

Step 0. $G_1 \succ G_2$

G_1: Minimize s_1^+ (Satisfy exposure goal)

G_2: Minimize s_2^- (Satisfy budget goal)

Step 1. Solve LP_1.

$$\text{Minimize } G_1 = s_1^+$$

subject to

$$\begin{aligned} 4x_1 + 8x_2 + s_1^+ - s_1^- &= 45 && \text{(Exposure goal)}\\ 8x_1 + 24x_2 + s_2^+ - s_2^- &= 100 && \text{(Budget goal)}\\ x_1 + 2x_2 &\le 10 && \text{(Personnel limit)}\\ x_1 &\le 6 && \text{(Radio limit)} \end{aligned}$$

$$x_1, x_2, s_1^+, s_1^-, s_2^+, s_2^- \ge 0$$

The optimum solution (determined by TORA) is $x_1 = 5$ minutes, $x_2 = 2.5$ minutes, $s_1^+ = 5$ million people, with the remaining variables equal to zero. The solution shows that the exposure goal, G_1, is violated by 5 million persons.

In LP_1, we have $\rho_1 = s_1^+$. Thus, the additional constraint we use with the G_2-problem is $s_1^+ = 5$.

Step 2. We need to solve LP_2, whose objective function is

$$\text{Minimize } G_2 = s_2^-$$

subject to the same set of constraints as in step 1 plus the additional constraint $s_1^+ = 5$. We can solve the new problem by using TORA's MODIFY option to add the constraint $s_1^+ = 5$.

The additional constraint $s_1^+ = 5$ can also be accounted for by substituting out s_1^+ in the first constraint. The result is that the right-hand side of the exposure goal constraint will be changed from 45 to 40, thus reducing LP_2 to

$$\text{Minimize } G_2 = s_2^-$$

subject to

$$\begin{aligned} 4x_1 + 8x_2 - s_1^- &= 40 && \text{(Exposure goal)}\\ 8x_1 + 24x_2 + s_2^+ - s_2^- &= 100 && \text{(Budget goal)}\\ x_1 + 2x_2 &\le 10 && \text{(Personnel limit)}\\ x_1 &\le 6 && \text{(Radio limit)} \end{aligned}$$

$$x_1, x_2, s_1^-, s_2^+, s_2^- \ge 0$$

The new formulation is one variable less than the one in LP_1, which is the general idea advanced by the column-dropping rule.

In reality, the optimization of LP_2 is not necessary in this example because the optimum solution to problem G_1 already yields $s_2^- = 0$. Hence, the solution of LP_1 is automatically optimum for LP_2 as well (you can verify this answer by solving LP_2 with TORA).

Next, we use an example to show that a better solution for the problem of Example 8.2-2 can be obtained if the preemptive method is used to *optimize* objectives rather than to *satisfy* goals. The example also serves to demonstrate the *column-dropping rule* for solving goal programs.

Example 8.2-3

The goals of Example 8.2-2 can be restated as

Priority 1: Maximize exposure (P_1)

Priority 2: Minimize cost (P_2)

Mathematically, the two objectives are given as

$$\begin{aligned} \text{Maximize } P_1 &= 4x_1 + 8x_2 && \text{(Exposure)}\\ \text{Minimize } P_2 &= 8x_1 + 24x_2 && \text{(Cost)} \end{aligned}$$

The specific goal limits for exposure and cost (= 45 and 100) are removed because the simplex method will determine them optimally.

The new problem can thus be stated as

$$\begin{aligned} \text{Maximize } P_1 &= 4x_1 + 8x_2\\ \text{Minimize } P_2 &= 8x_1 + 24x_2 \end{aligned}$$

subject to

$$\begin{aligned} x_1 + 2x_2 &\le 10\\ x_1 &\le 6\\ x_1, x_2 &\ge 0 \end{aligned}$$

We first solve the problem using the procedure introduced in Example 8.2-2.

Step 1. Solve LP_1.

$$\text{Maximize } P_1 = 4x_1 + 8x_2$$

subject to

$$\begin{aligned} x_1 + 2x_2 &\le 10\\ x_1 &\le 6\\ x_1, x_2 &\ge 0 \end{aligned}$$

The optimum solution (obtained by TORA) is $x_1 = 0$, $x_2 = 5$ with $P_1 = 40$, which shows that the most exposure we can get is 40 million persons.

Step 2. Add the constraint $4x_1 + 8x_2 \ge 40$ to ensure that goal G_1 is not degraded. Thus, we solve LP_2 as

$$\text{Minimize } P_2 = 8x_1 + 24x_2$$

subject to

$$x_1 + 2x_2 \leq 10$$
$$x_1 \leq 6$$
$$4x_1 + 8x_2 \geq 40 \quad \text{(Additional constraint)}$$
$$x_1, x_2 \geq 0$$

The TORA optimum solution of LP_2 is $P_2 = \$96{,}000$, $x_1 = 6$ minutes, and $x_2 = 2$ minutes. It yields the same exposure ($P_1 = 40$ million people) but at a smaller cost than the one in Example 8.2-2 where the main objective is to satisfy rather than optimize the goals.

The same problem is solved now by using the column-dropping rule. The rule calls for carrying the objective rows associated with all the goals in the simplex tableau.

LP_1 (Exposure Maximization): The LP_1 simplex tableau carries both objective rows, P_1 and P_2. The optimality condition applies to the P_1-objective row only. The P_2-row plays a passive role in LP_1, but must be updated with the rest of the simplex tableau in preparation for the optimization of LP_2.

LP_1 is solved in two iterations as follows:

Iteration	Basic	x_1	x_2	s_1	s_2	Solution
1	P_1	−4	−8	0	0	0
	P_2	−8	−24	0	0	0
	s_1	1	2	1	0	10
	s_2	1	0	0	1	6
2	P_1	0	0	4	0	40
	P_2	4	0	12	0	120
	x_2	$\frac{1}{2}$	1	$\frac{1}{2}$	0	5
	s_2	1	0	0	1	6

The last tableau yields the optimal solution $x_1 = 0$, $x_2 = 5$, and $P_1 = 40$.

The *column-dropping rule* calls for eliminating any *nonbasic* variable x_j with $z_j - c_j \neq 0$ from the optimum tableau of LP_1 before LP_2 is optimized. The reason for doing so is that these variables, if left unchecked, could become positive in lower priority optimization problems, which would degrade the quality of higher priority solutions.

LP_2 (Cost Minimization): The column-dropping rule eliminates s_1 (with $z_j - c_j = 4$). We can see from the P_2-row that if s_1 is not eliminated, it will be the entering variable at the start of the P_2-iterations and will yield the optimum solution $x_1 = x_2 = 0$, which will degrade the optimum objective value of the P_1-problem from $P_1 = 40$ to $P_1 = 0$. (Try it!)

The P_2-problem is of the minimization type. Following the elimination of s_1, the variable x_1 with $z_j - c_j = 4$ (>0) can improve the value of P_2. The following table shows the LP_2 iterations. The elements of P_1-row has been deleted because the row no longer serves a purpose in the optimization of LP_2.

Iteration	Basic	x_1	x_2	s_1	s_2	Solution
1	P_1					40
	P_2	4	0		0	120
	x_2	$\frac{1}{2}$	1		0	5
	s_2	1	0		1	6
2	P_1					40
	P_2	0	0		−4	96
	x_2	0	1		$-\frac{1}{2}$	2
	x_1	1	0		1	6

The optimum solution ($x_1 = 6$, $x_2 = 2$) with a total exposure of $P_1 = 40$ and a total cost of $P_2 = 96$ is the same as obtained earlier.

PROBLEM SET 8.2B

1. In Example 8.2-2, suppose that the budget goal is increased to \$110,000. The exposure goal remains unchanged at 45 million persons. Show how the preemptive method will reach a solution.
2. Solve Problem 1, Set 8.1a (Fairville tax model) using the following priority ordering for the goals: $G_1 \succ G_2 \succ G_3 \succ G_4 \succ G_5$.
3. Consider Problem 2, Set 8.1a, which deals with the presentation of band concerts and art shows at the NW Shopping Mall. Suppose that the goals set for teens, the young/middle-aged group, and seniors are referred to as G_1, G_2, and G_3, respectively. Solve the problem for each of the following priority orders:
   (a) $G_1 \succ G_2 \succ G_3$
   (b) $G_3 \succ G_2 \succ G_1$
   Show that the satisfaction of the goals (or lack of it) can be a function of the priority order.
4. Solve the Ozark University model (Problem 3, Set 8.1a) using the preemptive method, assuming that the goals are prioritized in the same order given in the problem.

SELECTED REFERENCES

Cohon, T. L., *Multiobjective Programming and Planning,* Academic Press, New York, 1978.

Ignizio, J. P., and T. M. Cavalier, *Linear Programming,* Prentice-Hall, Upper Saddle River, NJ, 1994.

Steuer, R. E., *Multiple Criteria Optimization: Theory, Computations, and Application,* Wiley, New York, 1986.

COMPREHENSIVE PROBLEMS

8.1[1] The Warehouzer Company manages three sites of forestland for timber production and reforestation with the respective areas of 100,000, 180,000, and 200,000 acres. The main

[1]Based on K. P. Rustagi, *Forest Management Planning for Timber Production: A Goal Programming Approach,* Bulletin No. 89, Yale University Press, New Haven, CT, 1976.

three events, and at least four of the team participants must be all-rounders. Set up an ILP model that can be used to select the competing team, and find the optimum solution using TORA.

9.3-6 In 1990, approximately 180,000 telemarketing centers employing 2 million individuals were in operation in the United States. In the year 2000, more than 700,000 companies employed approximately 8 million people in telemarketing their products. The questions of how many telemarketing centers to employ and where to locate them are of paramount importance.

The ABC company is in the process of deciding on the number of telemarketing centers to employ and their locations. A center may be located in one of several candidate areas selected by the company and may serve (partially or completely) one or more geographical areas. A geographical area is usually identified by one or more (telephone) area codes. ABC's telemarketing concentrates on eight area codes: 501, 918, 316, 417, 314, 816, 502, and 606. The following table provides the candidate locations, their served areas, and the cost of establishing the center.

Center location	Served area codes	Cost ($)
Dallas, TX	501, 918, 316, 417	500,000
Atlanta, GA	314, 816, 502, 606	800,000
Louisville, KY	918, 316, 417, 314, 816	400,000
Denver, CO	501, 502, 606	900,000
Little Rock, AR	417, 314, 816, 502	300,000
Memphis, TN	606, 501, 316, 417	450,000
St. Louis, MO	816, 502, 606, 314	550,000

Customers in all area codes can access any of the centers 24 hours a day.

The communication costs per hour between the centers and the area codes are given in the following table.

	From area code							
To	*501*	*918*	*316*	*417*	*314*	*816*	*502*	*606*
Dallas, TX	$14	$35	$29	$32	$25	$13	$14	$20
Atlanta, GA	$18	$18	$22	$18	$26	$23	$12	$15
Louisville, KY	$22	$25	$12	$19	$30	$17	$26	$25
Denver, CO	$24	$30	$19	$14	$12	$16	$18	$30
Little Rock, AR	$19	$20	$23	$16	$23	$11	$28	$12
Memphis, TN	$23	$21	$17	$21	$20	$23	$20	$10
St. Louis, MO	$17	$18	$12	$10	$19	$22	$16	$22

ABC would like to select between three and four centers. Where should they be located?