8

Networks, Distribution and PERT/CPM

8.1 What's Special About Network Models

A subclass of models called network LPs warrants special attention for three reasons:

1. They can be completely described by simple, easily understood graphical figures.
2. Under typical conditions, they have naturally integer answers, and one may find a network LP a useful device for describing and analyzing the various shipment strategies.
3. They are frequently easier to solve than general LPs.

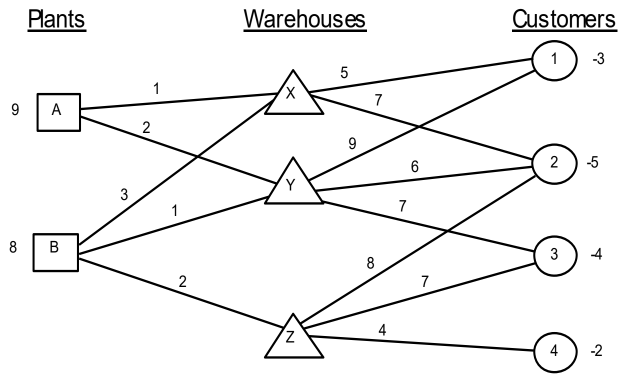
Physical examples that come to mind are pipeline or electrical transmission line networks. Any enterprise producing a product at several locations and distributing it to many warehouses and/or customers may find a network LP a useful device for describing and analyzing shipment strategies.

Although not essential, efficient specialized solution procedures may be used to solve network LPs. These procedures may be as much as 100 times faster than the general simplex method. Bradley, Brown, and Graves (1977) give a detailed description. Some of these specialized procedures were developed several years before the simplex method was developed for general LPs.

Figure 8.1 illustrates the network representing the distribution system of a firm using intermediate warehouses to distribute a product. The firm has two plants (denoted by *A* and *B*), three warehouses (denoted by *X*, *Y*, and *Z*), and four customer areas (denoted by 1, 2, 3, 4). The numbers adjacent to each node denote the availability of material at that node. Plant *A*, for example, has nine units available to be shipped. Customer 3, on the other hand, has –4 units meaning it needs to receive a shipment of four units.

The number above each arc is the cost per unit shipped along that arc. For example, if five of plant *A*'s nine units are shipped to warehouse *Y*, then a cost of $5 \times 2 = 10$ will be incurred as a direct result. The problem is to determine the amount shipped along each arc, so total costs are minimized and every customer has his requirements satisfied.

Figure 8.1 Three-Level Distribution Network



The essential condition on an LP for it to be a network problem is that it be representable as a network. There can be more than three levels of nodes, any number of arcs between any two nodes, and upper and lower limits on the amount shipped along a given arc.

With variables defined in an obvious way, the general LP describing this problem is:

```
[COST] MIN = AX + 2 * AY + 3 * BX + BY + 2 * BZ + 5 * X1
   + 7 * X2 + 9 * Y1 + 6 * Y2 + 7 * Y3 + 8 * Z2 + 7 * Z3
   + 4 * Z4;
[A] AX + AY <= 9;
[B] BX + BY + BZ <= 8;
[X] - AX - BX + X1 + X2 = 0;
[Y] - AY - BY + Y1 + Y2 + Y3 = 0;
[Z] - BZ + Z2 + Z3 + Z4 = 0;
[C1] - X1 - Y1 = -3;
[C2] - X2 - Y2 - Z2 = -5;
[C3] - Y3 - Z3 = -4;
[C4] - Z4 = -2;
```

There is one constraint for each node that is of a "sources = uses" form. Constraint 5, for example, is associated with warehouse *Y* and states that the amount shipped out minus the amount shipped in must equal 0.

A different view of the structure of a network problem is possible by displaying just the coefficients of the above constraints arranged by column and row. In the picture below, note that the apostrophes are placed every third row and column just to help see the regular patterns:

```
        A    A   B   B   B   X   X   Y   Y    Y    Z    Z    Z
        X    Y   X   Y   Z   1   2   1   2    3    2    3    4
COST:   1    2   3   1   2   5   7   9   6    7    8    7    4   MIN
A:      1    1  '           '            '             '         =  9
B:     '    '    1   1   1  '   '   '    '   '    '    '    '    =  8
X:     -1       -1           1   1       '             '         =
Y:          -1      -1      '        1   1    1        '         =
Z:     '    '   '   '   -1  '   '   '    '   '     1    1    1   =
C1:             '           -1      -1   '             '         = -3
C2:             '           '   -1      -1        -1   '         = -5
C3:    '    '   '   '   '   '   '   '    '   -1   '   -1   '    = -4
C4:             '           '            '             '    -1   = -2
```

You should notice the key feature of the constraint matrix of a network problem. That is, without regard to any bound constraints on individual variables, each column has exactly two nonzeroes in the constraint matrix. One of these nonzeroes is a +1, whereas the other is a −1. According to the convention we have adopted, the +1 appears in the row of the node from which the arc takes material, whereas the row of the node to which the arc delivers material is a −1. On a problem of this size, you should be able to deduce the optimal solution manually simply from examining Figure 8.1. You may check it with the computer solution below:

```
Variable           Value        Reduced Cost
      AX        3.000000            0.000000
      AY        3.000000            0.000000
      BX        0.000000            3.000000
      BY        6.000000            0.000000
      BZ        2.000000            0.000000
      X1        3.000000            0.000000
      X2        0.000000            0.000000
      Y1        0.000000            5.000000
      Y2        5.000000            0.000000
      Y3        4.000000            0.000000
      Z2        0.000000            3.000000
      Z3        0.000000            1.000000
      Z4        2.000000            0.000000
     Row    Slack or Surplus       Dual Price
    COST      100.000000           -1.000000
       A        3.000000            0.000000
       B        0.000000            1.000000
       X        0.000000            1.000000
       Y        0.000000            2.000000
       Z        0.000000            3.000000
      C1        0.000000            6.000000
      C2        0.000000            8.000000
      C3        0.000000            9.000000
      C4        0.000000            7.000000
```

This solution exhibits two pleasing features found in the solution to any network problem:

1. If the right-hand side coefficients (the capacities and requirements) are integer, then the variables will also be integer.
2. If the objective coefficients are integer, then the dual prices will also be integer.

We can summarize network LPs as follows:

1. Associated with each node is a number that specifies the amount of commodity available at that node (negative implies that commodity is required.)
2. Associated with each arc are:
   a) a cost per unit shipped (which may be negative) over the arc,
   b) a lower bound on the amount shipped over the arc (typically zero), and
   c) an upper bound on the amount shipped over the arc (infinity in our example).

The problem is to determine the flows that minimize total cost subject to satisfying all the supply, demand, and flow constraints.

8.1.1 Special Cases

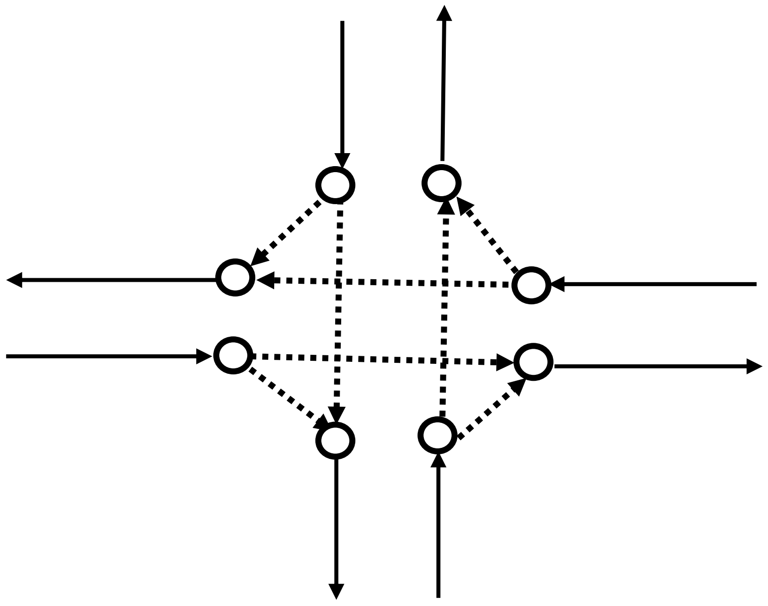
There are a number of common applications of LP models that are special cases of the standard network LP. The ones worthy of mention are:

1. *Transportation or distribution problems*. A two-level network problem, where all the nodes at the first level are suppliers, all the nodes at the second level are users, and the only arcs are from suppliers to users, is called a transportation, or distribution model.
2. *Shortest and longest path problems*. Suppose one is given the road network of the United States and wishes to find the shortest route from Bangor to San Diego. This is equivalent to a special case of a network or transshipment problem in which one unit of material is available at Bangor and one unit is required at San Diego. The cost of shipping over an arc is the length of the arc. Simple, fast procedures exist for solving this problem. An important first cousin of this problem, the longest route problem, arises in the analysis of PERT/CPM projects.
3. *The assignment problem*. A transportation problem in which the number of suppliers equals the number of customers, each supplier has one unit available, and each customer requires one unit, is called an assignment problem. An efficient, specialized procedure exists for its solution.
4. *Maximal flow*. Given a directed network with an upper bound on the flow on each arc, one wants to find the maximum that can be shipped through the network from some specified origin, or source node, to some other destination, or sink node. Applications might be to determine the rate at which a building can be evacuated or military material can be shipped to a distant trouble spot.

8.1.2 Fitting into Network Structure: Roads with No Left Turns

The parcel delivery service, UPS got publicity a number of years ago when it claimed that its drivers did not make left turns. The argument is that at a busy intersection, making a left turn requires the left turning vehicle to wait for a gap in oncoming traffic. We illustrate here that it sometimes requires a bit of thought to precisely describe a problem as a network problem. How do we represent restrictions on turns at an intersection, left turns and U turns in particular? One way of representing turn restrictions in a standard directed network is to add arcs and nodes to an intersection to represent the valid possibilities. Figure 8.2 illustrates what one can do to represent a 4-way intersection where left turns and U turns are prohibited. One node is replaced by 8 nodes and 8 additional arcs. Observe that for a Roundabout, however, no additional nodes and arcs are required. Some UPS drivers have admitted that left turns are sometimes made, usually only on streets with low traffic.

Figure 8.2 Network for a 4-Way Intersection with No Left or U Turns



8.2 PERT/CPM Networks and LP

Program Evaluation and Review Technique (PERT) and Critical Path Method (CPM) are two closely related techniques for monitoring the progress of a large project. A key part of PERT/CPM is calculating the critical path. That is, identifying the subset of the activities that must be performed exactly as planned in order for the project to finish on time.

We will show that the calculation of the critical path is a very simple network LP problem, specifically, a longest path problem. You do not need this fact to efficiently calculate the critical path, but it is an interesting observation that becomes useful if you wish to examine a multitude of "crashing" options for accelerating a tardy project.

In the table below, we list the activities involved in the simple, but nontrivial, project of building a house. An activity cannot be started until all of its predecessors are finished:

Activity	Mnemonic	Activity Time	Predecessors (Mnemonic)
Dig Basement	DIG	3	—
Pour Foundation	FOUND	4	DIG
Pour Basement Floor	POURB	2	FOUND
Install Floor Joists	JOISTS	3	FOUND
Install Walls	WALLS	5	FOUND
Install Rafters	RAFTERS	3	WALLS, POURB
Install Flooring	FLOOR	4	JOISTS
Rough Interior	ROUGH	6	FLOOR
Install Roof	ROOF	7	RAFTERS
Finish Interior	FINISH	5	ROUGH, ROOF
Landscape	SCAPE	2	POURB, WALLS

In Figure 8.3, we show the so-called PERT (or activity-on-arrow) network for this project. We would like to calculate the minimum elapsed time to complete this project. Relative to this figure, the number of interest is simply the longest path from left to right in this figure. The project can be completed no sooner than the sum of the times of the successive activities on this path. Verify for yourself that the critical path consists of activities *DIG*, *FOUND*, *WALLS*, *RAFTERS*, *ROOF*, and *FINISH* and has length 27.

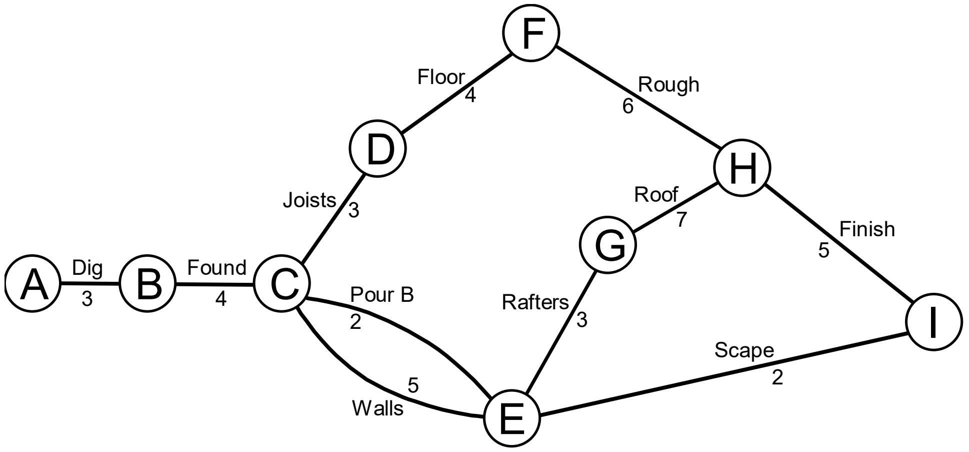
Even though this example can be worked out by hand, almost without pencil and paper, let us derive an LP formulation for solving this problem. Most people attempting this derivation will come up with one of two seemingly unrelated formulations.

The first formulation is motivated as follows. Let variables *DIG*, *FOUND*, etc. be either 1 or 0 depending upon whether activities *DIG*, *FOUND*, etc. are on or not on the critica1 path. The variables equa1 to one will define the critical path. The objective function will be related to the fact that we want to find the maximum length path in the PERT diagram.

Our objective is in fact:

```
MAX = 3 * DIG + 4 * FOUND + 2 * POURB + 3 * JOISTS +
      5 * WALLS + 3 * RAFTERS + 4 * FLOOR + 6 *
      ROUGH + 7 * ROOF + 5 * FINISH + 2 * SCAPE;
```

Figure 8.3 Activity-on-Arc PERT/CPM Network



By itself, this objective seems to take the wrong point of view. We do not want to maximize the project length. However, if we specify the proper constraints, we shall see this objective will seek out the maximum length path in the PERT network. We want to use the constraints to enforce the following:

1. *DIG* must be on the critical path.
2. An activity can be on the critical path only if one of its predecessors is on the critical path. Further, if an activity is on a critical path, exactly one of its successors must be on the critical path, if it has successors.
3. Exactly one of *SCAPE* or *FINISH* must be on the critical path.

Convince yourself the following set of constraints will enforce the above:

```
– DIG = –1;
– FOUND + DIG = 0;
– JOISTS — POURB — WALLS + FOUND = 0;
– FLOOR + JOISTS = 0;
– RAFTERS – SCAPE + POURB + WALLS = 0;
– ROUGH + FLOOR = 0;
– ROOF + RAFTERS = 0;
– FINISH + ROUGH + ROOF = 0;
+ FINISH + SCAPE = +1;
```

If you interpret the length of each arc in the network as the scenic beauty of the arc, then the formulation corresponds to finding the most scenic route by which to ship one unit from *A* to *I*.

The solution of the problem is:

```
Optimal solution found at step:         2
Objective value:                 27.00000

Variable           Value        Reduced Cost
     DIG        1.000000          0.0000000
   FOUND        1.000000          0.0000000
   POURB       0.0000000           3.000000
  JOISTS       0.0000000          0.0000000
   WALLS        1.000000          0.0000000
 RAFTERS        1.000000          0.0000000
   FLOOR       0.0000000          0.0000000
   ROUGH       0.0000000           2.000000
    ROOF        1.000000          0.0000000
  FINISH        1.000000          0.0000000
   SCAPE       0.0000000           13.00000

     Row    Slack or Surplus      Dual Price
       1        27.00000           1.000000
       2       0.0000000           6.000000
       3       0.0000000          -9.000000
       4       0.0000000          -5.000000
       5       0.0000000          -2.000000
       6       0.0000000          0.0000000
       7       0.0000000           2.000000
       8       0.0000000           3.000000
       9       0.0000000           10.00000
      10       0.0000000           15.00000
```

Notice the variables corresponding to the activities on the critical path have a value of 1. What is the solution if the first constraint, –*DIG* = –1, is deleted?

It is instructive to look at the PICTURE of this problem in the following figure:

```
                           R
                   J       A               F
        F   P   O   W   F   F   R       I   S
        O   O   I   A   T   L   O   R   N   C
    D   U   U   S   L   E   O   U   O   I   A
    I   N   R   T   L   R   O   G   O   S   P
    G   D   B   S   S   S   R   H   F   H   E

 1:   3   4   2   3   5   3   4   6   7   5   2   MAX
 2:  -1           '           '           '       = -1
 3:   1  -1   '   '   '   '   '   '   '   '   '   =
 4:       1  -1  -1  -1       '           '       =
 5:               1          -1           '       =
 6:   '   '   1   '   1  -1   '   '   '   '  -1   =
 7:               '           1  -1       '       =
 8:               '       1   '      -1   '       =
 9:   '   '   '   '   '   '   '   1   1  -1   '   =
10:               '           '           1   1   = 1
```

Notice that each variable has at most two coefficients in the constraints. When two, they are +1 and –1. This is the distinguishing feature of a network LP.

Now, let us look at the second possible formulation. The motivation for this formulation is to minimize the elapsed time of the project. To do this, realize that each node in the PERT network represents an event (e.g., as follows: *A*, start digging the basement; *C*, complete the foundation; and *I*, complete landscaping and finish interior).

Define variables *A*, *B*, *C*, …, *H*, *I* as the time at which these events occur. Our objective function is then:

```
MIN = I – A;
```

These event times are constrained by the fact that each event has to occur later than each of its preceding events, at least by the amount of any intervening activity. Thus, we get one constraint for each activity:

```
B – A >=  3;      ! DIG;
C – B >=  4;      ! FOUND;
E – C >=  2;
D – C >=  3;
E – C >=  5;
F – D >=  4;
G – E >=  3;
H – F >=  6;
H – G >=  7;
I – H >=  5;
I – E >=  2;
```

The solution to this problem is:

```
Optimal solution found at step:         0
Objective value:                 27.00000
Variable           Value        Reduced Cost
       I        27.00000           0.0000000
       A       0.0000000           0.0000000
       B        3.000000           0.0000000
       C        7.000000           0.0000000
       E        12.00000           0.0000000
       D        10.00000           0.0000000
       F        14.00000           0.0000000
       G        15.00000           0.0000000
       H        22.00000           0.0000000
     Row    Slack or Surplus      Dual Price
       1        27.00000            1.000000
       2       0.0000000           -1.000000
       3       0.0000000           -1.000000
       4        3.000000           0.0000000
       5       0.0000000           0.0000000
       6       0.0000000           -1.000000
       7       0.0000000           0.0000000
       8       0.0000000           -1.000000
       9        2.000000           0.0000000
      10       0.0000000           -1.000000
      11       0.0000000           -1.000000
      12        13.00000           0.0000000
```

Notice that the objective function value equals the critical path length. We can indirectly identify the activities on the critical path by noting the constraints with nonzero dual prices. The activities corresponding to these constraints are on the critical path. This correspondence makes sense. The right-hand side of a constraint is the activity time. If we increase the time of an activity on the critical path, it should increase the project length and thus should have a nonzero dual price. What is the solution if the first variable, *A*, is deleted?

The PICTURE of the coefficient matrix for this problem follows:

```
    A  B  C  D  E  F  G  H  I
 1:-1        '        '     1 MIN
 2:-1  1     '        '       > 3
 3: ' -1 '1  '    '   '    '  > 4
 4:      -1  '  1     '       > 2
 5:      -1  1        '       > 3
 6: '    -1  '  1 '   '    '  > 5
 7:         -1     1  '       > 4
 8:          ' -1     1       > 3
 9: '    '   '    -1  '  1 '  > 6
10:          '       -1  1    > 7
11:          '        ' -1  1 > 5
12: '    '   ' -1 '   '    '1 > 2
```

Notice the PICTURE of this formulation is essentially the PICTURE of the previous formulation rotated ninety degrees. Even though these two formulations originally were seemingly unrelated, there is really an incestuous relationship between the two, a relationship that mathematicians politely refer to as duality.

8.3 Activity-on-Arc vs. Activity-on-Node Network Diagrams

Two conventions are used in practice for displaying project networks: (1) Activity-on-Arc (AOA) and (2) Activity-on-Node (AON). Our previous example used the AOA convention. The characteristics of the two are:

AON

- Each activity is represented by a node in the network.
- A precedence relationship between two activities is represented by an arc or link between the two.
- AON may be less error prone because it does not need dummy activities or arcs.

AOA

- Each activity is represented by an arc in the network.
- If activity *X* must precede activity *Y,* there are *X* leads into arc *Y*. The nodes thus represent events or "milestones" (e.g., "finished activity *X*"). Dummy activities of zero length may be required to properly represent precedence relationships.
- AOA historically has been more popular, perhaps because of its similarity to Gantt charts used in scheduling.

An AON project with six activities is shown in Figure 8.4. The number next to each node is the duration of the activity. Activities A and B are the sources or start of the project. Activity F is the final activity. By inspection, you can discover that the longest path consists of activities *A*, *C*, *E*, and *F*. It has a length of 29. The corresponding AOA network for the same project is shown in Figure 8.5. In the AOA network, we have enclosed the activity letters in circles above the associated arc. The unenclosed numbers below each arc are the durations of the activities. We have given the nodes, or milestones, arbitrary number designations enclosed in squares. Notice the dummy activity (the dotted arc) between nodes 3 and 4. This is because *a dummy activity will be required in an AOA diagram anytime that two activities* (e.g., *A* and *B*) *share some* (e.g., activity *D*), *but not all* (e.g., activity *C*), *successor activities.*

Figure 8.4 An Activity-on-Node Representation

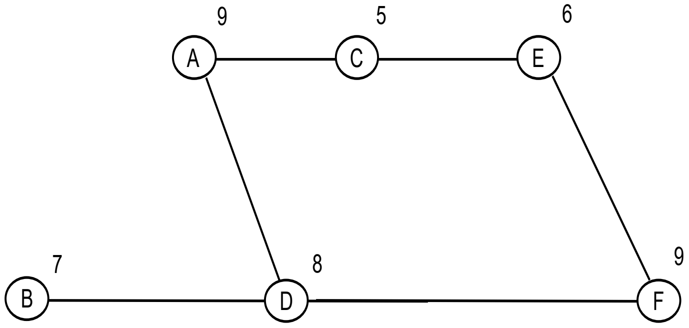
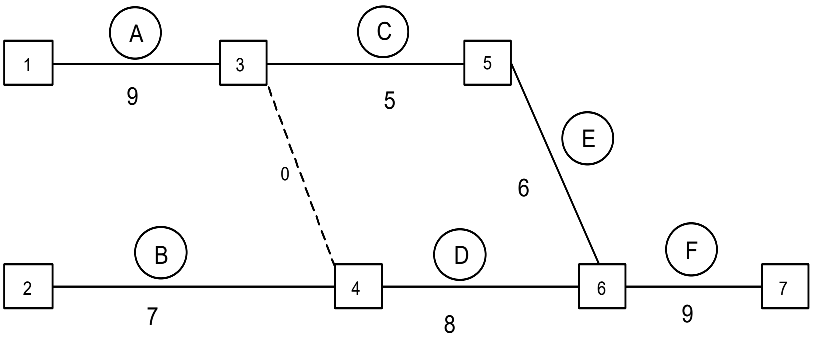


Figure 8.5 An Activity-on-Arc Representation



8.4 Crashing of Project Networks

Once the critical path length for a project has been identified, the next question invariably asked is: can we shorten the project? The process of decreasing the duration of a project or activity is commonly called crashing. For many construction projects, it is common for the customer to pay an incentive to the contractor for finishing the project in a shorter length of time. For example, in highway repair projects, it is not unusual to have incentives from $5,000 to $25,000 per day that the project is finished before a target date.

8.4.1 The Cost and Value of Crashing

There is value in crashing a project. In order to crash a project, we must crash one or more activities. Crashing an activity costs money. Deciding to crash an activity requires us to compare the cost of crashing that activity with the value of the resulting reduction in project length. This decision is frequently complicated by the fact that some negotiation may be required between the party that incurs the cost of crashing the activity (e.g., the contractor) and the party that enjoys the value of the crashed project (e.g., the customer).

8.4.2 The Cost of Crashing an Activity

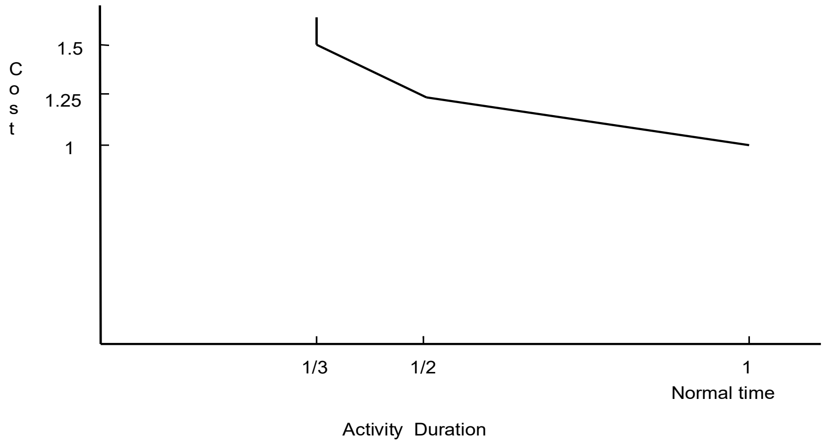
An activity is typically crashed by applying more labor to it (e g., overtime or a second shift). We might typically expect that using second-shift labor could cost 1.5 times as much per hour as first-shift labor. We might expect third-shift labor to cost twice as much as first-shift labor.

Consider an activity that can be done in six days if only first-shift labor is used and has a labor cost of \$6,000. If we allow the use of second-shift labor and thus work two shifts per day, the activity can be done in three days for a cost of $3 \times 1000 + 3 \times 1000 \times 1.5 = 7{,}500$. If third-shift labor is allowed, then the project can be done in two days by working three shifts per day and incurring a total of:

$$2 \times 1000 + 2 \times 1000 \times 1.5 + 2 \times 1000 \times 2 = \$9{,}000.$$

Thus, we get a crashing cost curve for the activity as shown in Figure 8.6:

Figure 8.6 Activity Crash Cost Curve



8.4.3 The Value of Crashing a Project

There are two approaches to deciding upon the amount of project crashing: (a) we simply specify a project duration time and crash enough to achieve this duration, or (b) we estimate the value of crashing it for various days. As an example of (a), in 1987 a new stadium was being built for the Montreal Expos baseball team. The obvious completion target was the first home game of the season.

As an example of (b), consider an urban expressway repair. What is the value per day of completing it early? Suppose that 6,000 motorists are affected by the repair project and each is delayed by 10 minutes each day because of the repair work (e.g., by taking alternate routes or by slower traffic). The total daily delay is 6,000 × 10 = 60,000 minutes = 1000 hours. If we assign an hourly cost of \$5/person × hours, the social value of reducing the repair project by one day is \$5,000.

8.4.4 Formulation of the Crashing Problem

Suppose we have investigated the crashing possibilities for each activity or task in our previous project example. These estimates are summarized in the following table:

Activity	Predecessor	Normal duration (Days)	Minimum duration if crashed (Days)	\$/Day
A	—	9	5	5000
B	—	7	3	6000
C	A	5	3	4000
D	A,B	8	4	2000
E	C	6	3	3000
F	D,E	9	5	9000

For example, activity *A* could be done in five days rather than nine. However, this would cost us an extra (9 – 5) × 5000 = \$20,000.

First, consider the simple case where we have a hard due date by which the project must be done. Let us say 22 days in this case. How would we decide which activities to crash? Activity *D* is the cheapest to crash per day. However, it is not on the critical path, so its low cost is at best just interesting.

Let us define:

EF_i = earliest finish time of activity *i*, taking into account any crashing that is done;
C_i = number of days by which activity *i* is crashed.

In words, the LP model will be:

Minimize Cost of crashing
subject to
For each activity *j* and each predecessor *i*:
earliest finish of *j* ≥ earliest finish of predecessor *i* + actual duration of *j*;
For each activity *j:*
minimum duration for *j* if crashed ≤ actual duration of *j* ≤ normal duration for *j*.

A LINGO formulation is:

```
! Find optimal crashing for a project with a due date;
SETS:
 TASK: NORMAL, FAST, COST, EF, ACTUAL;
 PRED( TASK, TASK)::
ENDSETS
```

```
DATA:
  TASK, NORMAL, FAST, COST =
   A       9     5    5000
   B       7     3    6000
   C       5     3    4000
   D       8     4    2000
   E       6     3    3000
   F       9     5    9000;
  PRED =
   A, C
   A, D
   B, D
   C, E
   D, F
   E, F;
 DUEDATE = 22;
ENDDATA
!-------------------------------------;
! Minimize the cost of crashing;
 [OBJ] MIN = @SUM( TASK( I): COST( I)*( NORMAL( I) - ACTUAL( I)));
! For tasks with no predecessors...;
 @FOR( TASK( J): EF( J) >= ACTUAL( J););
!   and for those with predecessors;
 @FOR( PRED( I, J):
    EF( J) >= EF( I) + ACTUAL( J);
     );
! Bound the actual time;
 @FOR( TASK( I):
     @BND( FAST(I), ACTUAL( I), NORMAL( I));
     );
! Last task is assumed to be last in project;
  EF( @SIZE( TASK)) <= DUEDATE;
```

Part of the solution is:

```
Global optimal solution found at step:            24
 Objective value:                             31000.00
        Variable           Value        Reduced Cost
          EF( A)        7.000000           0.0000000
          EF( B)        7.000000           0.0000000
          EF( C)        10.00000           0.0000000
          EF( D)        13.00000           0.0000000
          EF( E)        13.00000           0.0000000
          EF( F)        22.00000           0.0000000
      ACTUAL( A)        7.000000           0.0000000
      ACTUAL( B)        7.000000           -4000.000
      ACTUAL( C)        3.000000            1000.000
      ACTUAL( D)        6.000000           0.0000000
      ACTUAL( E)        3.000000            2000.000
      ACTUAL( F)        9.000000           -2000.000
```

Thus, for an additional cost of $31,000, we can meet the 22-day deadline.

Now, suppose there is no hard project due date, but we do receive an incentive payment of $5000 for each day we reduce the project length. Define *PCRASH* = number of days the project is finished before the twenty-ninth day. Now, the formulation is:

```
! Find optimal crashing for a project with
   a due date and incentive for early completion;
SETS:
 TASK: NORMAL, FAST, COST, EF, ACTUAL;
 PRED( TASK, TASK):;
ENDSETS
DATA:
  TASK, NORMAL, FAST, COST =
   A        9      5     5000
   B        7      3     6000
   C        5      3     4000
   D        8      4     2000
   E        6      3     3000
   F        9      5     9000;
  PRED =
   A, C
   A, D
   B, D
   C, E
   D, F
   E, F;
! Incentive for each day we beat the due date;
 INCENT = 5000;
 DUEDATE = 29;
ENDDATA
!--------------------------------------;
! Minimize the cost of crashing
     less early completion incentive payment;
  [OBJ] MIN = @SUM( TASK( I): COST( I)*( NORMAL( I) - ACTUAL( I)))
            - INCENT * PCRASH;

! For tasks with no predecessors...;
 @FOR( TASK( J): EF( J) >= ACTUAL( J););
!   and for those with predecessors;
 @FOR( PRED( I, J):
    EF( J) >= EF( I) + ACTUAL( J);
     );
! Bound the actual time;
 @FOR( TASK( I):
     @BND( FAST(I), ACTUAL( I), NORMAL( I));
     );
! Last task is assumed to be last in project;
  EF( @SIZE( TASK)) + PCRASH = DUEDATE;
```

From the solution, we see we should crash it by five days to give a total project length of twenty-four days:

```
  Global optimal solution found at step:            21
   Objective value:                               -6000.000
              Variable            Value        Reduced Cost
                PCRASH         5.000000           0.0000000
                EF( A)         7.000000           0.0000000
                EF( B)         7.000000           0.0000000
                EF( C)         12.00000           0.0000000
                EF( D)         15.00000           0.0000000
                EF( E)         15.00000           0.0000000
                EF( F)         24.00000           0.0000000
            ACTUAL( A)         7.000000           0.0000000
            ACTUAL( B)         7.000000           -6000.000
            ACTUAL( C)         5.000000           -1000.000
            ACTUAL( D)         8.000000           0.0000000
            ACTUAL( E)         3.000000           0.0000000
            ACTUAL( F)         9.000000           -4000.000
```

The excess of the incentive payments over crash costs is $6,000.

8.5 Resource Constraints in Project Scheduling

For many projects, a major complication is that there are a limited number of resources. The limited resources require you to do tasks individually that otherwise might be done simultaneously. Pritzker, Watters, and Wolfe (1969) gave a formulation representing resource constraints in project and jobshop scheduling problems. The formulation is based on the following key ideas: a) time is discrete rather than continuous (e.g., each period is a day), b) for every activity and every discrete period there is a 0/1 variable that is one if that activity starts in that period, and c) for every resource and period there is a constraint that enforces the requirement that the amount of resource required in that period does not exceed the amount available.

To illustrate, we take the example considered previously with shorter activity times, so the total number of periods is smaller:

```
     MODEL:
     ! PERT/CPM project scheduling with resource constraints(PERTRSRC);
     ! There is a limited number of each resource/machine.
     ! An activity cannot be started until: 1) all its predecessors have
completed, and  2) resources/machines required are available.;

        SETS:
     ! There is a set of tasks with a given duration, and
         a start time to be determined;
        TASK: TIME, START, ES;
     ! The precedence relations, the first task in the
         precedence relationship needs to be completed before the
         second task can be started;
        PRED( TASK, TASK);
     ! There are a set of periods;
        PERIOD;
```

```
   RESOURCE: CAP;
! Some operations need capacity in some department;
   TXR( TASK, RESOURCE): NEED;
! SX( I, T) = 1 if task I starts in period T;
   TXP( TASK, PERIOD): SX;
   RXP( RESOURCE, PERIOD);
  ENDSETS

 DATA:
 ! Upper limit on number of periods required to complete the project;
   PERIOD = 1..20;
 ! Task names and duration;
   TASK  TIME =
   FIRST    0
   FCAST    7
   SURVEY   2
   PRICE    1
   SCHED    3
   COSTOUT  2
   FINAL    4;

! The predecessor/successor combinations;
  PRED=  FIRST,FCAST,    FIRST,SURVEY,
         FCAST,PRICE,    FCAST,SCHED,    SURVEY,PRICE,
         SCHED,COSTOUT,  PRICE,FINAL,    COSTOUT,FINAL;
! There are 2 departments, accounting and operations,
   with capacities...;
   RESOURCE = ACDEPT, OPNDEPT;
        CAP =   1,       1;
! How much each task needs of each resource;
          TXR,       NEED =
     FCAST,  OPNDEPT,  1
     SURVEY, OPNDEPT,  1
     SCHED,  OPNDEPT,  1
     PRICE,   ACDEPT,  1
     COSTOUT, ACDEPT,  1;
 ENDDATA
!------------------------------------------------------------;
! Minimize start time of last task;
 MIN = START( @SIZE( TASK));
! Start time for each task;
 @FOR( TASK( I):
   [DEFSTRT] START( I) = @SUM( PERIOD( T): T * SX( I, T));
     );
 @FOR( TASK( I):
!  Each task must be started in some period;
   [MUSTDO]  @SUM( PERIOD( T): SX( I, T)) = 1;
! The SX vars are binary, i.e., 0 or 1;
   @FOR( PERIOD( T): @BIN( SX( I, T)););
      );
! Precedence constraints;
  @FOR( PRED( I, J):
    [PRECD]  START( J) >= START( I) + TIME( I);
```

```
         );
   ! Resource usage, For each resource R and period T;
     @FOR( RXP( R, T):
   ! Sum over all tasks I that use resource R in period T;
       [RSRUSE] @SUM( TXR( I, R):
          @SUM( PERIOD( S)| S #GE# ( T - ( TIME( I) - 1)) #AND# S #LE# T:
                  NEED( I, R) * SX( I, S))) <= CAP( R);
              );
   ! The following makes the formulation tighter;
   ! Compute earliest start disregarding resource constraints;
     @FOR( TASK( J):
       ES( J) = @SMAX( 0, @MAX( PRED( I, J): ES( I) + TIME(I)));
      ! Task cannot start earlier than unconstrained early start;
       @SUM( PERIOD(T) | T #LE# ES( J): SX( J, T)) = 0;
         );
   END
```

When solved, we get a project length of 14. If there were no resource constraints, then the project length would be 13:

```
Global optimal solution found
 Objective value:       14.00000

       Variable           Value
  START( FIRST)        1.000000
  START( FCAST)        1.000000
 START( SURVEY)        11.00000
  START( PRICE)        13.00000
  START( SCHED)        8.000000
START( COSTOUT)        11.00000
  START( FINAL)        14.00000
```

8.6 Path Formulations

In many network problems, it is natural to think of a solution in terms of paths that material takes as it flows through the network. For example, in Figure 8.1, there are thirteen possible paths. Namely:

$$A \to X \to 1,\ A \to X \to 2,\ A \to Y \to 1,\ A \to Y \to 2,\ A \to Y \to 3,\ B \to X \to 1,\ B \to X \to 2,$$
$$B \to Y \to 1,\ B \to Y \to 2,\ B \to Y \to 3,\ B \to Z \to 2,\ B \to Z \to 3,$$
$$B \to Z \to 4$$

One can, in fact, formulate decision variables in terms of complete paths rather than just simple links, where the path decision variable corresponds to using a combination of links. This is a form of what is sometimes called a composite variable approach. The motivations for using the path approach are:

1. More complicated cost structures can be represented. For example, Geoffrion and Graves (1974) use the path formulation to represent “milling in transit” discount fare structures in shipping food and feed products.

2. Path-related restrictions can be incorporated. For example, regulations allow a truck driver to be on duty for at most 10 hours. Thus, in a truck routing network one would not consider paths longer than 10 hours. In a supply chain network, a path that is long may be prohibited because it may cause lead times to be too long.
3. The number of rows (constraints) in the model may be substantially less.
4. In integer programs where some, but not all, of the problem has a network structure, the path formulation may be easier to solve.

8.6.1 Example

Let us reconsider the first problem (Figure 8.1, page 148). Suppose shipments from *A* to *X* are made by the same carrier as shipments from *X* to 2. This carrier will give a $1 per unit "milling-in-transit" discount for each unit it handles from both *A* to *X* and *X* to 2. Further, the product is somewhat fragile and cannot tolerate a lot of transportation. In particular, it cannot be shipped both over link $B \rightarrow X$ and $X \rightarrow 2$ or both over links $A \rightarrow Y$ and $Y \rightarrow 1$.

Using the notation *AX1* = number of units shipped from *A* to *X* to 1, etc., the path formulation is:

```
MIN =  6 * PAX1 + 7 * PAX2  + 8 * PAY2
     + 9 * PAY3 + 8 * PBX1 + 10 * PBY1
     + 7 * PBY2 + 8 * PBY3 + 10 * PBZ2
     + 9 * PBZ3 + 6 * PBZ4;
  [A]   PAX1 + PAX2 + PAY2 + PAY3 <= 9;
  [B]   PBX1 + PBY1 + PBY2 + PBY3
     + PBZ2 + PBZ3 + PBZ4 <= 8;
 [C1]   PAX1 + PBX1 + PBY1 = 3;
 [C2]   PAX2 + PAY2 + PBY2 + PBZ2 = 5;
 [C3]   PAY3 + PBY3 + PBZ3 = 4;
 [C4]   PBZ4 = 2;
```

Notice the cost of path *AX2* = 1 + 7 – 1 = 7. In addition, paths *BX2* and *AY1* do not appear. This model has only six constraints as opposed to nine in the original formulation. The reduction in constraints arises from the fact that, in path formulations, one does not need the "sources = uses" constraints for intermediate nodes.

In general, the path formulation will have fewer rows, but more decision variables than the corresponding network LP model.

When we solve, we get:

```
Objective value=  97.0000
   Variable          Value
       PAX1       3.000000
       PAX2       3.000000
       PBY2       2.000000
       PBY3       4.000000
       PBZ4       2.000000
```

This is cheaper than the previous solution, because the three units shipped over path *AX2* go for $1 per unit less.

A path formulation need not have a naturally integer solution. If the path formulation, however, is equivalent to a network LP, then it will have a naturally integer solution.

The path formulation is popular in long-range forest planning. See, for example, Davis and Johnson (1986), where it is known as the "Model I" approach. The standard network LP based formulation is known as the "Model II" approach. In a forest planning Model II, a link in the network represents a decision to plant an acre of a particular kind of tree in some specified year and harvest it in some future specified year. A node represents a specific harvest and replant decision. A decision variable in Model I is a complete prescription of how to manage (i.e., harvest and replant) a given piece of land over time. Some Model I formulations in forest planning may have just a few hundred constraints, but over a million decision variables or paths.

There is a generalization of the path formulation to arbitrary linear programs, known as Fourier/Motzkin/Dines elimination, see for example Martin (1999) and Dantzig (1963). The transformation of a network LP to the path formulation involves eliminating a particular node (constraint), by generating a new variable for every combination of input arc and output arc incident to the node. A constraint in an arbitrary LP can be eliminated if it is first converted to a constraint with a right-hand side of zero and then a new variable is generated for every combination of positive and negative coefficient in the constraint. The disadvantage of this approach is that even though the number of constraints is reduced to one, the number of variables may grow exponentially with the number of original constraints.

A variable corresponding to a path in a network is an example of a composite variable, a general approach that is sometimes useful for representing complicated/ing constraints. A composite variable is one that represents a feasible combination of two or more original variables. The complicating constraints are represented implicitly by generating only those composite variables that correspond to feasible combinations and values of the original variables.

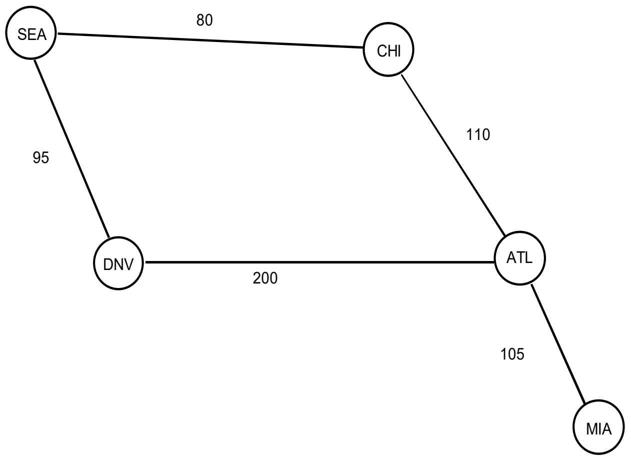
8.7 Path Formulations of Undirected Networks

In many communications networks, the arcs have capacity, but are undirected. For example, when you are carrying on a phone conversation with someone in a distant city, the conversation uses capacity on all the links in your connection. However, you cannot speak of a direction of flow of the connection.

A major concern for a long distance communications company is the management of its communications network. This becomes particularly important during certain holidays, such as Mother's Day, Thanksgiving, Yippe Hakkah, etc. Not only does the volume of calls increase on these days, but also the pattern of calls changes dramatically from the business-oriented traffic during weekdays in the rest of the year. A communications company faces two problems: (a) the design problem. That is, what capacity should be installed on each link? As well as, (b) the operations problem. That is, given the installed capacity, how are demands best routed? The path formulation is an obvious format for modeling an undirected network. The following illustrates the operational problem.

Consider the case of a phone company with the network structure shown in Figure 8.7:

Figure 8.7 Phone Company Network Structure



The number next to each arc is the number of calls that can be in progress simultaneously along that arc. If someone in *MIA* tries to call his mother in *SEA*, the phone company must first find a path from *MIA* to *SEA* such that each arc on that path is not at capacity. It is quite easy to inefficiently use the capacity. Suppose there is a demand for 110 calls between *CHI* and *DNV* and 90 calls between *ATL* and *SEA*. Further, suppose all of these calls were routed over the *ATL*, *DNV* link. Now, suppose we wish to make a call between *MIA* and *SEA*. Such a connection is impossible because every path between the two contains a saturated link (i.e., either *ATL*, *DNV* or *CHI*, *ATL*). However, if some of the 110 calls between *CHI* and *DNV* were routed over the *CHI*, *SEA*, *DNV* links, then one could make calls between *MIA* and *SEA*. In conventional voice networks, a call cannot be rerouted once it has started. In packet switched data networks and, to some extent, in cellular phone networks, some rerouting is possible.

8.7.1 Example

Suppose during a certain time period the demands in the table below occur for connections between pairs of cities:

	DNV	**CHI**	**ATL**	**MIA**
SEA	10	20	38	33
DNV		42	48	23
CHI			90	36
ATL				26

Which demands should be satisfied and via what routes to maximize the number of connections satisfied?

Solution. If we use the path formulation, there will be two paths between every pair of cities except *ATL* and *MIA*. We will use the notation $P1_{ij}$ for number of calls using the shorter or more northerly path between cities i and j, and $P2_{ij}$ for the other path, if any. There will be two kinds of constraints:

1) a capacity constraint for each link, and
2) an upper limit on the calls between each pair of cities, based on available demand.

A formulation is:

```
! Maximize calls carried;
MAX = P1MIAATL + P1MIADNV + P2MIADNV
    + P1MIASEA + P2MIASEA + P1MIACHI
    + P2MIACHI + P1ATLDNV + P2ATLDNV
    + P1ATLSEA + P2ATLSEA + P1ATLCHI
    + P2ATLCHI + P1DNVSEA + P2DNVSEA
    + P1DNVCHI + P2DNVCHI + P1SEACHI
    + P2SEACHI;
! Capacity constraint for each link;
[KATLMIA]  P1MIAATL + P1MIADNV + P2MIADNV
     + P1MIASEA + P2MIASEA + P1MIACHI
     + P2MIACHI <= 105;
[KATLDNV]  P1MIADNV + P1MIASEA + P1MIACHI
     + P1ATLDNV + P1ATLSEA + P1ATLCHI
     + P2DNVSEA + P2DNVCHI + P2SEACHI <= 200;
[KDNVSEA]  P2MIADNV + P1MIASEA + P1MIACHI
     + P2ATLDNV + P1ATLSEA + P1ATLCHI
     + P1DNVSEA + P1DNVCHI + P2SEACHI <=  95;
[KSEACHI]  P2MIADNV + P2MIASEA + P1MIACHI
     + P2ATLDNV + P2ATLSEA + P1ATLCHI
     + P2DNVSEA + P1DNVCHI + P1SEACHI <=  80;
[KATLCHI]  P2MIADNV + P2MIASEA + P2MIACHI
     + P2ATLDNV + P2ATLSEA + P2ATLCHI
     + P2DNVSEA + P2DNVCHI + P2SEACHI <= 110;
! Demand constraints for each city pair;
[DMIAATL]             P1MIAATL <= 26;
[DMIADNV]  P1MIADNV + P2MIADNV <= 23;
[DMIASEA]  P1MIASEA + P2MIASEA <= 33;
[DMIACHI]  P1MIACHI + P2MIACHI <= 36;
[DATLDNV]  P1ATLDNV + P2ATLDNV <= 48;
[DATLSEA]  P1ATLSEA + P2ATLSEA <= 38;
[DATLCHI]  P1ATLCHI + P2ATLCHI <= 90;
[DDNVSEA]  P1DNVSEA + P2DNVSEA <= 10;
[DDNVCHI]  P1DNVCHI + P2DNVCHI <= 42;
[DSEACHI]  P1SEACHI + P2SEACHI <= 20;
```

When this formulation is solved, we see we can handle 322 out of the total demand of 366 calls:

```
Optimal solution found at step:        11
Objective value:                 322.0000
Variable           Value        Reduced Cost
P1MIAATL        26.00000            0.000000
P1MIADNV        23.00000            0.000000
P2MIADNV         0.00000            2.000000
P1MIASEA         0.00000            0.000000
P2MIASEA         0.00000            0.000000
P1MIACHI        25.00000            0.000000
P2MIACHI         0.00000            0.000000
P1ATLDNV        48.00000            0.000000
P2ATLDNV         0.00000            2.000000
P1ATLSEA        38.00000            0.000000
P2ATLSEA         0.00000            0.000000
P1ATLCHI        23.00000            0.000000
P2ATLCHI        67.00000            0.000000
P1DNVSEA         3.50000            0.000000
P2DNVSEA         6.50000            0.000000
P1DNVCHI         5.50000            0.000000
P2DNVCHI        36.50000            0.000000
P1SEACHI        20.00000            0.000000
P2SEACHI         0.00000            2.000000
     Row    Slack or Surplus      Dual Price
       1       322.00000            1.000000
 KATLMIA        31.00000            0.000000
 KATLDNV         0.00000            0.000000
 KDNVSEA         0.00000            1.000000
 KSEACHI         0.00000            0.000000
 KATLCHI         0.00000            1.000000
 DMIAATL         0.00000            1.000000
 DMIADNV         0.00000            1.000000
 DMIASEA        33.00000            0.000000
 DMIACHI        11.00000            0.000000
 DATLDNV         0.00000            1.000000
 DATLSEA         0.00000            0.000000
 DATLCHI         0.00000            0.000000
 DDNVSEA         0.00000            0.000000
 DDNVCHI         0.00000            0.000000
 DSEACHI         0.00000            1.000000
```

Verify that the demand not carried is *MIA-CHI*: 11 and *MIA-SEA*: 33. Apparently, there are a number of alternate optima.

8.8 Double Entry Bookkeeping: A Network Model of the Firm

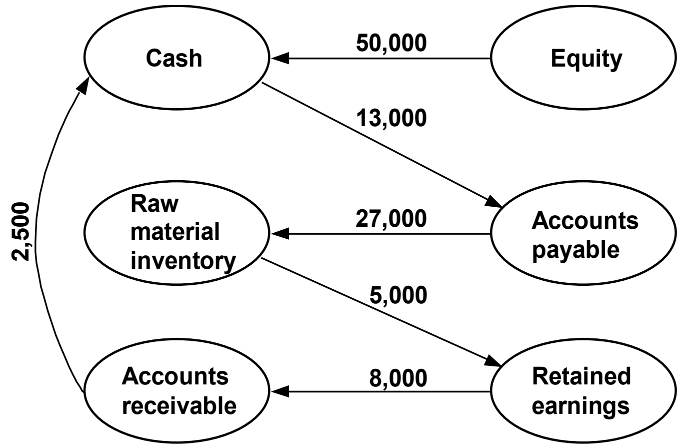
Authors frequently like to identify who was the first to use a given methodology. A contender for the distinction of formulating the first network model is Fra Luca Pacioli. In 1594, while director of a Franciscan monastery in Italy, he published a description of the accounting convention that has come to be known as double entry bookkeeping. From the perspective of networks, each double entry is an arc in a network.

To illustrate, suppose you start up a small dry goods business. During the first two weeks, the following transactions occur:

CAP	1)	You invest $50,000 of capital in cash to start the business.
UR	2)	You purchase $27,000 of product on credit from supplier *S*.
PAY	3)	You pay $13,000 of your accounts payable to supplier *S*.
SEL	4)	You sell $5,000 of product to customer *C* for $8,000 on credit.
REC	5)	Customer *C* pays you $2,500 of his debt to you.

In our convention, liabilities and equities will typically have negative balances. For example, the initial infusion of cash corresponds to a transfer (an arc) from the equity account (node) to the cash account, with a flow of $50,000. The purchase of product on credit corresponds to an arc from the accounts payable account node to the raw materials inventory account, with a flow of $27,000. Paying $13,000 to the supplier corresponds to an arc from the cash account to the accounts payable account, with a flow of $13,000. Figure 8.8 illustrates.

Figure 8.8 Double Entry Bookkeeping as a Network Model



8.9 Extensions of Network LP Models

There are several generalizations of network models that are important in practice. These extensions share two features in common with true network LP models, namely:

- They can be represented graphically.
- Specialized, fast solution procedures exist for several of these generalizations.

The one feature typically not found with these generalizations is:

- Solutions are usually not naturally integer, even if the input data are integers.

The important generalizations we will consider are:

1. *Networks with Gains*. Sometimes called generalized networks, this generalization allows a specified gain or loss of material as it is shipped from one node to another. Structurally, these problems are such that every column has at most two nonzeroes in the constraint matrix. However, the requirement that these coefficients be +1 and −1 is relaxed. Specialized procedures, which may be twenty times faster than the regular simplex method, exist for solving these problems.

   Examples of "shipments" with such gains or losses are: investment in an interest-bearing account, electrical transmission with loss, natural gas pipeline shipments where the pipeline pumps burn natural gas from the pipeline, and work force attrition. Stroup and Wollmer (1992) show how a network with gains model is useful in the airline industry for deciding where to purchase fuel and where to ferry fuel from one stop to another. Truemper (1976) points out, if the network with gains has no circuits when considered as an undirected network, then it can be converted to a pure network model by appropriate scaling.
2. *Undirected Networks*. In communications networks, there is typically no direction of shipment. The arcs are undirected.
3. *Multicommodity Networks.* In many distribution situations, there are multiple commodities moving through the network, all competing for scarce network capacity. Each source may produce only one of the commodities and each destination, or sink, may accept only one specific commodity.
4. *Leontief Flow*. In a so-called Leontief input-output model (see Leontief, 1951), each activity uses several commodities although it produces only one commodity. For example, one unit of automotive production may use a half ton of steel, 300 pounds of plastic, and 100 pounds of glass. Material Requirements Planning (MRP) models have the same feature. If each output required only one input, then we would simply have a network with gains. Special purpose algorithms exist for solving Leontief Flow and MRP models. See, for example, Jeroslow, Martin, Rardin, and Wang (1992).
5. *Activity/Resource Diagrams*. If Leontief flow models are extended, so each activity can have not only several inputs, but also several outputs, then one can in fact represent arbitrary LPs. We call the obvious extension of the network diagrams to this case an activity/resource diagram.

8.9.1 Multicommodity Network Flows

In a network LP, one assumption is a customer is indifferent, except perhaps for cost, to the source from which his product was obtained. Another assumption is that there is a single commodity flowing through the network. In many network-like situations, there are multiple distinct commodities flowing through the network. If each link has infinite capacity, then an independent network flow LP could be solved for each commodity. However, if a link has a finite capacity that applies to the sum of all commodities flowing over that link, then we have a multicommodity network problem.

The most common setting for multicommodity network problems is in shipping. The network might be a natural gas pipeline network and the commodities might be different fuels shipped over the network. In other shipping problems, such as traffic assignment or overnight package delivery, each origin/destination pair constitutes a commodity.

The crucial feature is identity of the commodities must be maintained throughout the network. That is, customers care which commodity gets delivered. An example is a metals supply company that ships aluminum bars, stainless steel rings, steel beams, etc., all around the country, using a single limited capacity fleet of trucks.

In general form, the multicommodity network problem is defined as:

D_{ik} = demand for commodity k at node i, with negative values denoting supply;
C_{ijk} = cost per unit of shipping commodity k from node i to node j;
U_{ij} = capacity of the link from node i to node j.

We want to find:

X_{ijk} = amount of commodity k shipped from node i to node j, so as to:

$$\min \quad \sum_i \sum_j \sum_k c_{ijk}\, x_{ijk}$$

subject to:

For each commodity k and node t :

$$\sum_i x_{itk} = D_{tk} + \sum_j x_{tjk}$$

For each link i, j:

$$\sum_k x_{ijk} \le U_{ij}$$

8.9.2 Reducing the Size of Multicommodity Problems

If the multiple commodities correspond to origin destination pairs and the cost of shipping a unit over a link is independent of the final destination, then you can aggregate commodities over destinations. That is, you need identify a commodity only by its 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 only 100 commodities, rather than 10,000 commodities.

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

8.9.3 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. The matrix below represents the average number of tons potential customers need to move between each origin/destination pair per day. For example, people in city 2 need to move four 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 *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. That is, *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 goods maintain their identity as they move through the network. Notice 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 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 for the commodity. Define:

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

The resulting formulation is:

```
MODEL:
! Keywords: multi-commodity, network flow, routing;
! Multi-commodity network flow problem;
SETS:
! The nodes in the network;
   NODES/1..6/:;
! The set of nodes that are origins;
   COMMO(NODES)/1, 2, 5/:;
   EDGES(NODES, NODES): D, C, U, V;
   NET(EDGES, COMMO): X;
ENDSETS
DATA:
! Demand: amount to be shipped from
   origin(row) to destination(col);
D = 0 5 9 7 0 4
    0 0 4 0 1 0
    0 0 0 0 0 0
    0 0 0 0 0 0
    0 4 0 2 0 8
    0 0 0 0 0 0;
! Cost per unit shipped over a arc/link;
C = 0 4 5 8 9 9
    3 0 3 2 4 6
    5 3 0 2 3 5
    7 3 3 0 5 6
    8 5 3 6 0 3
    9 7 4 5 5 0;
! Upper limit on amount shipped on each link;
U = 0 2 3 2 1 20
    0 0 2 8 3 9
    3 0 0 1 3 9
    5 4 6 0 5 9
    1 0 2 7 0 9
    9 9 9 9 9 0;
```

```
! Whether an arc/link exists or not;
! V = 0 if U = 0;
! V = 1 otherwise;
V = 0 1 1 1 1 1
    0 0 1 1 1 1
    1 0 0 1 1 1
    1 1 1 0 1 1
    1 0 1 1 0 1
    1 1 1 1 1 0;
ENDDATA
! Minimize shipping cost over all links;
MIN = @SUM( NET(I, J, K): C(I, J) * X(I, J, K));
! This is the balance constraint. There are two cases:
 Either the node that needs to be balanced is not a supply,
 in which case the sum of incoming amounts
 minus the sum of outgoing amounts must equal
 the demand for that commodity for that city;
!or where the node is a supply,
 the sum of incoming minus outgoing amounts must equal
 the negative of the sum of the demand for the commodity
 that the node supplies;
   @FOR(COMMO(K): @FOR(NODES(J)|J #NE# K:
      @SUM(NODES(I): V(I, J) * X(I, J, K) - V(J, I) * X(J, I, K))
        = D(K, J);
       );
   @FOR(NODES(J)|J #EQ# K:
     @SUM(NODES(I): V(I, J) * X(I, J, K) - V(J, I) * X(J, I, K))
      = -@SUM( NODES(L): D(K, L)););
! This is a capacity constraint;
   @FOR(EDGES(I, J)|I #NE# J:
      @SUM(COMMO(K): X(I, J, K)) <= U(I, J);
        );
END
```

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

```
Optimal solution found at step:        56
Objective value:                 361.0000

   Variable            Value        Reduced Cost
X( 1, 2, 1)         2.000000          0.0000000
X( 1, 3, 1)         3.000000          0.0000000
X( 1, 4, 1)         2.000000          0.0000000
X( 1, 5, 1)         1.000000          0.0000000
X( 1, 6, 1)         17.00000          0.0000000
X( 2, 3, 2)         2.000000          0.0000000
X( 2, 4, 2)         2.000000          0.0000000
X( 2, 5, 2)         1.000000          0.0000000
X( 3, 4, 5)         1.000000          0.0000000
X( 4, 2, 5)         4.000000          0.0000000
X( 4, 3, 2)         2.000000          0.0000000
```

```
X( 5, 3, 1)        1.000000          0.0000000
X( 5, 3, 5)        1.000000          0.0000000
X( 5, 4, 5)        5.000000          0.0000000
X( 5, 6, 5)        8.000000          0.0000000
X( 6, 2, 1)        3.000000          0.0000000
X( 6, 3, 1)        5.000000          0.0000000
X( 6, 4, 1)        5.000000          0.0000000
```

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

8.9.4 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 each vehicle takes. This is sometimes called the FTL(Full Truck Load) routing problem. The assignment part is of interest if the firm has several different fleets of vehicles available. Then the question is 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. A similar approach has been used at US Airways by Kontogiorgis and Acharya (1999).

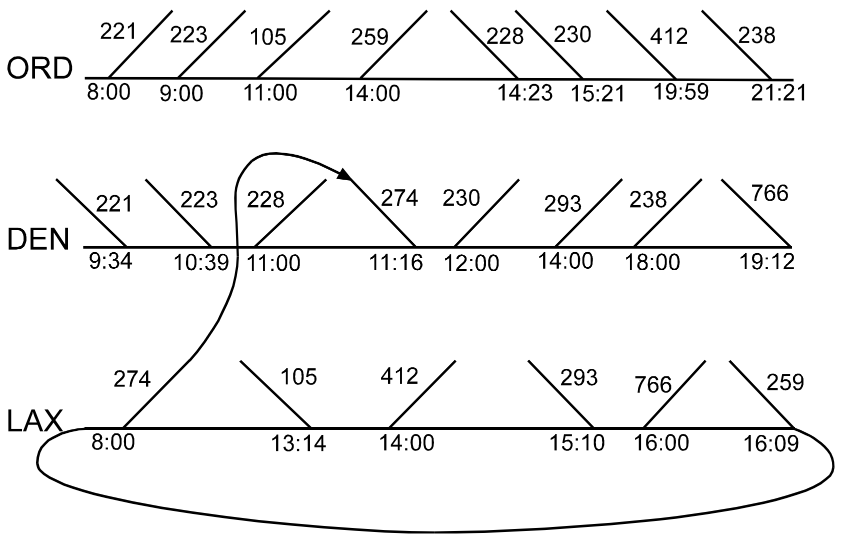
To motivate things, consider the following set of flights serving Chicago (ORD), Denver (DEN), and Los Angeles (LAX) that United Airlines once offered on a typical weekday:

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

This schedule can be represented by the network in Figure 8.9. The diagonal lines from upper left to lower right represent flight arrivals. The diagonal lines from lower left to upper right represent departures. To complete the diagram, we need to add the lines connecting each flight departure to each flight arrival. The thin line connecting the departure of Flight 274 from LAX to the arrival of Flight 274 in Denver illustrates one of the missing lines. If the schedule repeats every day, it is reasonable to have the network have a backloop for each city, as illustrated for LAX. To avoid clutter, these lines have not been added.

Figure 8.9 A Fleet Routing Network



Perhaps the obvious way of interpreting this as a network problem is as follows:

a) Each diagonal line (with the connection to its partner) constitutes a variable, corresponding to a flight;
b) each horizontal line or backloop corresponds to a decision variable representing the number of aircraft on the ground;
c) each point of either an arrival or a departure constitutes a node, and the model will have a constraint saying, in words:

   (no. of aircraft on the ground at this city at this instant) + (arrivals at this instant) = (no. of departures from this city at this instant) + (no. of aircraft on the ground after this instant).

With this convention, there would be 22 constraints (8 at *ORD*, 8 at *DEN*, and 6 at *LAX*), and 33 variables (11 flight variables and 22 ground variables). The number of constraints and variables can be reduced substantially if we make the observation that the feasibility of a solution is not affected if, for each city:

a) Each arrival is delayed until the first departure after that arrival.
b) Each departure is advanced (made earlier) to the most recent departure just after an arrival.

Thus, the only nodes required are when a departure immediately follows an arrival.

If we have a fleet of just one type of aircraft, we probably want to know what is the minimum number of aircrafts needed to fly this schedule. In words, our model is:

Minimize number of aircraft on the ground overnight
(That is the only place they can be, given the flight schedule)
subject to
   source of aircraft = use of aircraft at each node of the network
   and each flight must be covered.

Taking all the above observations into account gives the following formulation of a network LP. Note the *G* variables represent the number of aircraft on the ground at a given city just after a specified instant:

```
! Fleet routing with a single plane type;
!  Minimize number of planes on ground overnight;
MIN = GC2400 + GD2400 + GL2400;
! The plane(old) conservation constraints;
! Chicago at 8 am, sources - uses = 0;
GC2400 - F221 - F223 - F105 - F259 - GC1400 = 0;
! Chicago at midnight;
GC1400 + F228 + F230 + F412 + F238 - GC2400 = 0;
! Denver at 11 am;
GD2400 + F221 + F223 - F228 - GD1100 = 0;
! Denver at high noon;
GD1100 + F274 - F230 - F293 - F238 - GD1800 = 0;
! Denver at midnight;
GD1800 + F766 - GD2400 = 0;
! LA at 8 am;
GL2400 - F274 - GL0800 = 0;
! LA at 1400;
GL0800 + F105 - F412 - GL1400 = 0;
! LA at 1600;
GL1400 + F293 - F766 - GL1600 = 0;
! LA at midnight;
GL1600 + F259 - GL2400 = 0;
! Cover our flight's constraints;
 F221 = 1;
 F223 = 1;
 F274 = 1;
 F105 = 1;
 F228 = 1;
 F230 = 1;
 F259 = 1;
 F293 = 1;
 F412 = 1;
 F766 = 1;
 F238 = 1;
```

This model assumes no deadheading is used. That is, no plane is flown empty from one city to another in order to position it for the next day. The reader probably figured out by simple intuitive arguments that six aircraft are needed. The following solution gives the details:

```
Optimal solution found at step:         0
Objective value:                 6.000000

Variable           Value        Reduced Cost
  GC2400        4.000000           0.0000000
  GD2400        1.000000           0.0000000
  GL2400        1.000000           0.0000000
    F221        1.000000           0.0000000
    F223        1.000000           0.0000000
    F105        1.000000           0.0000000
    F259        1.000000           0.0000000
  GC1400       0.0000000            1.000000
    F228        1.000000           0.0000000
    F230        1.000000           0.0000000
    F412        1.000000           0.0000000
    F238        1.000000           0.0000000
  GD1100        2.000000           0.0000000
    F274        1.000000           0.0000000
    F293        1.000000           0.0000000
  GD1800       0.0000000            1.000000
    F766        1.000000           0.0000000
  GL0800       0.0000000           0.0000000
  GL1400       0.0000000           0.0000000
  GL1600       0.0000000            1.000000
```

Thus, there are four aircraft on the ground overnight at Chicago, one overnight at Denver, and one overnight at Los Angeles.

8.9.5 Fleet Assignment

If we have two or more aircraft types, then we have the additional decision of specifying the type of aircraft assigned to each flight. The typical setting is we have a limited number of new aircraft that are more efficient than previous aircraft. Let us extend our previous example by assuming we have two aircraft of type B. They are more fuel-efficient than our original type A aircraft. However, the capacity of type B is slightly less than A. We now probably want to maximize the profit contribution. The profit contribution from assigning an aircraft of type *i* to flight *j* is:

+ (revenue from satisfying all demand on flight *j*)
– (“spill” cost of not being able to serve all demand on *j* because of the limited capacity of aircraft type *i*)
– (the operating cost of flying aircraft type *i* on flight *j*)
+ (revenue from demand spilled from previous flights captured on this flight).

The spill costs and recoveries are probably the most difficult to estimate.

The previous model easily generalizes with the two modifications:

a) Conservation of flow constraints is needed for each aircraft type.
b) The flight coverage constraints become more flexible, because there are now two ways of covering a flight.

After carefully calculating the profit contribution for each combination of aircraft type and flight, we get the following model:

```
! Fleet routing and assignment with two plane types;
! Maximize profit contribution from flights covered;
MAX = 105 * F221A + 121 * F221B + 109 * F223A + 108
      * F223B + 110 * F274A + 115 * F274B + 130 *
      F105A + 140 * F105B + 106 * F228A + 122 *
      F228B + 112 * F230A + 115 * F230B + 132 *
      F259A + 129 * F259B + 115 * F293A + 123 *
      F293B + 133 * F412A + 135 * F412B + 108 *
      F766A + 117 * F766B + 116 * F238A + 124 *
      F238B;
! Conservation of flow constraints;
! for type A aircraft;
! Chicago at 8 am, sources - uses = 0;
F221A - F223A - F105A - F259A - GC1400A + GC2400A=0;
! Chicago at midnight;
F228A + F230A + F412A + F238A + GC1400A - GC2400A=0;
! Denver at 11 am;
   F221A + F223A - F228A - GD1100A + GD2400A = 0;
! Denver at high noon;
F274A - F230A - F293A - F238A + GD1100A - GD1800A=0;
! Denver at midnight;
   F766A - GD2400A + GD1800A = 0;
! LA at 8 am;
   - F274A - GL0800A + GL2400A = 0;
! LA at 1400;
   F105A - F412A + GL0800A - GL1400A = 0;
! LA at 1600;
   F293A - F766A + GL1400A - GL1600A = 0;
! LA at midnight;
   F259A - GL2400A + GL1600A = 0;
! Aircraft type B, conservation of flow;
! Chicago at 8 am;
-F221B - F223B - F105B - F259B - GC1400B +GC2400B=0;
! Chicago at midnight;
F228B + F230B + F412B + F238B + GC1400B - GC2400B=0;
! Denver at 11 am;
   F221B + F223B - F228B - GD1100B + GD2400B = 0;
! Denver at high noon;
F274B - F230B - F293B - F238B + GD1100B - GD1800B=0;
! Denver at midnight;
   F766B - GD2400B + GD1800B = 0;
! LA at 8 am;
   - F274B - GL0800B + GL2400B = 0;
! LA at 1400;
   F105B - F412B + GL0800B - GL1400B = 0;
! LA at 1600;
   F293B - F766B + GL1400B - GL1600B = 0;
! LA at midnight;
   F259B - GL2400B + GL1600B = 0;
! Can put at most one plane on each flight;
```

```
   F221A + F221B <= 1;
   F223A + F223B <= 1;
   F274A + F274B <= 1;
   F105A + F105B <= 1;
   F228A + F228B <= 1;
   F230A + F230B <= 1;
   F259A + F259B <= 1;
   F293A + F293B <= 1;
   F412A + F412B <= 1;
   F766A + F766B <= 1;
   F238A + F238B <= 1;
! Fleet size of type B;
   GC2400B + GD2400B + GL2400B <= 2;
```

The not so obvious solution is:

```
Optimal solution found at step:        37
Objective value:                 1325.000

Variable           Value        Reduced Cost
   F221B        1.000000           0.0000000
   F223A        1.000000           0.0000000
   F274A        1.000000           0.0000000
   F105A        1.000000           0.0000000
   F228B        1.000000           0.0000000
   F230A        1.000000           0.0000000
   F259A        1.000000           0.0000000
   F293B        1.000000           0.0000000
   F412A        1.000000           0.0000000
   F766B        1.000000           0.0000000
   F238A        1.000000           0.0000000
 GC2400A        3.000000           0.0000000
 GD1100A        1.000000           0.0000000
 GL2400A        1.000000           0.0000000
 GC2400B        1.000000           0.0000000
 GD1100B        1.000000           0.0000000
 GD2400B        1.000000           0.0000000
```

Six aircraft are still used. The newer type *B* aircraft cover flights 221, 228, 293, and 766. Since there are two vehicle types, this model is a multicommodity network flow model rather than a pure network flow model. Thus, we are not guaranteed to be able to find a naturally integer optimal solution to the LP. Nevertheless, such was the case for the example above.

Generating an explicit model as above would be tedious. The following is a set-based version of the above model. With the set based version, adding a flight or an aircraft type is a fairly simple clerical operation:

```
MODEL:
SETS:  ! Fleet routing and assignment (FLEETRAV);
 CITY :;  ! The cities involved;
 ACRFT:   ! Aircraft types;
  FCOST,  !  Fixed cost per day of this type;
  FSIZE;  !  Max fleet size of this type;
 FLIGHT:;
 FXCXC( FLIGHT, CITY, CITY) :
  DEPAT,  ! Flight departure time;
  ARVAT;  ! arrival time at dest.;
 AXC( ACRFT, CITY):
  OVNITE; ! Number staying overnite by type,city;
 AXF( ACRFT, FXCXC):
  X,      ! Number aircraft used by type,flight;
  PC;     ! Profit contribution by type,flight;
ENDSETS
DATA:
 CITY = ORD  DEN  LAX;
 ACRFT, FCOST, FSIZE =
  MD90     0    7
  B737     0    2;
 FLIGHT = F221 F223 F274 F105 F228 F230 F259 F293 F412 F766 F238;
 FXCXC, DEPAT, ARVAT =
!     Flight  Origin Dest. Depart Arrive;
         F221    ORD    DEN     800     934
         F223    ORD    DEN     900    1039
         F274    LAX    DEN     800    1116
         F105    ORD    LAX    1100    1314
         F228    DEN    ORD    1100    1423
         F230    DEN    ORD    1200    1521
         F259    ORD    LAX    1400    1609
         F293    DEN    LAX    1400    1510
         F412    LAX    ORD    1400    1959
         F766    LAX    DEN    1600    1912
         F238    DEN    ORD    1800    2121;
 PC =   ! Profit contribution of each vehicle*flight combo;
   105        109        110         130         106          112
   132        115        133         108         116
   121        108        115         140         122          115
   129        123        135         117         124;
ENDDATA
!------------------------------------------------------------------;
! Maximize profit contribution from flights minus
   overhead cost of aircraft in fleet;
 MAX = @SUM( AXF( I, N, J, K): PC( I, N, J, K) * X( I, N, J, K))
     - @SUM( AXC( I, J): FCOST( I) * OVNITE( I, J));
```

```
! At any instant, departures in particular, the number of
 cumulative arrivals must be >= number of cumulative departures;
! For each flight of each aircraft type;
 @FOR( ACRFT( I):
  @FOR( FXCXC( N, J, K):
! Aircraft on ground in morning +
   number aircraft arrived thus far >=
   number aircraft departed thus far;
   OVNITE( I, J) +
   @SUM( FXCXC( N1, J1, K1)| K1 #EQ# J #AND#
                            ARVAT( N1, J1, K1) #LT# DEPAT( N, J, K):
               X( I, N1, J1, J)) >=
   @SUM( FXCXC( N1, J1, K1)| J1 #EQ# J #AND#
                            DEPAT( N1, J1, K1) #LE# DEPAT( N, J, K):
               X( I, N1, J, K1));
         );););
! This model does not allow deadheading, so at the end of the day,
   arrivals must equal departures;
 @FOR( ACRFT( I):
   @FOR( CITY( J):
    @SUM( AXF( I, N, J1, J): X( I, N, J1, J)) =
    @SUM( AXF( I, N, J, K): X( I, N, J, K));
       );
     );
!  Each flight must be covered;
    @FOR( FXCXC( N, J, K):
       @SUM( AXF( I, N, J, K): X( I, N, J, K)) = 1;
        );
! Fleet size limits;
   @FOR( ACRFT( I):
     @SUM( AXC( I, J): OVNITE( I, J)) <= FSIZE( I);
       );
! Fractional planes are not allowed;
   @FOR( AXF: @GIN( X); );
END
```

Sometimes, especially in trucking, one has the option of using rented vehicles to cover only selected trips. With regard to the model, the major modification is that rented vehicles do not have to honor the conservation of flow constraints. Other details that are sometimes included relate to maintenance. With aircraft, for example, a specific aircraft must be taken out of service for maintenance after a specified number of landings, or after a specified number of flying hours, or after a certain elapsed time, whichever occurs first. It is not too difficult to incorporate such details, although the model becomes substantially larger.

8.9.6 Leontief Flow Models

In a Leontief flow model, each activity produces one output. However, it may use zero or more inputs. The following example illustrates.

Example: Islandia Input-Output Model

The country of Islandia has four major export industries: steel, automotive, electronics, and plastics. The economic minister of Islandia would like to maximize exports-imports. The unit of exchange in Islandia is the klutz. The prices in klutzes on the world market per unit of steel, automotive, electronics, and plastics are, respectively: 500, 1500, 300, and 1200. Production of one unit of steel requires 0.02 units of automotive production, 0.01 units of plastics, 250 klutzes of raw material purchased on the world market, plus one-half man-year of labor. Production of one automotive unit requires 0.8 units of steel, 0.15 units of electronics, 0.11 units of plastic, one man-year of labor, and 300 klutzes of imported material. Production of one unit of electronic equipment requires 0.01 units of steel, 0.01 units of automotive, 0.05 units of plastic, half a man-year of labor, and 50 klutzes of imported material. Automotive production is limited at 650,000 units. Production of one unit of plastic requires 0.03 units of automotive production, 0.2 units of steel, 0.05 units of electronics, 2 man-years of labor, plus 300 klutzes of imported materials. The upper limit on plastic is 60,000 units. The total manpower available in Islandia is 830,000 men per year. No steel, automotive, electronics, or plastic products may be imported.

How much should be produced and exported of the various products?

Formulation and Solution of the Islandia Problem

The formulation of an input-output model should follow the same two-step procedure for formulating any LP model. Namely, (1) identify the decision variables and (2) identify the constraints. The key to identifying the decision variables for this problem is to make the distinction between the amount of commodity produced and the amount exported. Once this is done, the decision variables can be represented as:

PROD(STEEL)	=	units of steel produced,
PROD(AUTO)	=	units of automotive produced,
PROD(PLASTIC)	=	units of plastic produced,
PROD(ELECT)	=	units of electronics produced,
EXP(STEEL)	=	units of steel exported,
EXP(AUTO)	=	units of automotive exported,
EXP(PLASTIC)	=	units of plastic exported,
EXP(ELECT)	=	units of electronics exported.

The commodities can be straightforwardly identified as steel, automotive, electronics, plastics, manpower, automotive capacity, and plastics capacity. Thus, there will be seven constraints.

The sets formulation and solution are:

```
MODEL: ! Islandia Input/output model;
SETS:
 COMMO:
   PROD, EXP, REV, COST, MANLAB, CAP;
 CXC(COMMO, COMMO): USERATE;
ENDSETS
DATA:
  COMMO = STEEL, AUTO, PLASTIC, ELECT;
   COST =    250    300     300      50;
   REV  =    500   1500    1200     300;
 MANLAB =    .5     1       2        .5;
! Amount used of the column commodity per unit
   of the row commodity;
 USERATE=    -1    .02     .01        0
             .8     -1     .11       .15
             .2    .03      -1       .05
             .01   .01     .05        -1;
  MANPOWER = 830000;
  CAP =    999999 650000  60000 999999;
ENDDATA
[PROFIT] MAX = @SUM( COMMO: REV * EXP - PROD * COST);
 @FOR( COMMO( I):
   [ NETUSE] ! Net use must equal = 0;
     EXP(I) + @SUM(COMMO(J): USERATE(J,I)* PROD(J))
     = 0;
   [CAPLIM] PROD( I) <= CAP( I);
     );
 [MANLIM] @SUM(COMMO:PROD * MANLAB) < MANPOWER;
END
```

Notice this model has the Leontief flow feature. Namely, each decision variable has at most one negative constraint coefficient.

The solution is:

```
Global optimal solution found.
Objective value:    0.4354312E+09

        Variable           Value        Reduced Cost
        MANPOWER        830000.0            0.000000
    PROD( STEEL)        393958.3            0.000000
     PROD( AUTO)        475833.3            0.000000
  PROD( PLASTIC)        60000.00            0.000000
    PROD( ELECT)        74375.00            0.000000
     EXP( STEEL)        547.9167            0.000000
      EXP( AUTO)        465410.4            0.000000
   EXP( PLASTIC)        0.000000            2096.875
     EXP( ELECT)        0.000000            121.8750
     REV( STEEL)        500.0000            0.000000
      REV( AUTO)        1500.000            0.000000
   REV( PLASTIC)        1200.000            0.000000
```

```
                     REV( ELECT)        300.0000              0.000000
                    COST( STEEL)        250.0000              0.000000
                     COST( AUTO)        300.0000              0.000000
                  COST( PLASTIC)        300.0000              0.000000
                    COST( ELECT)        50.00000              0.000000
                  MANLAB( STEEL)       0.5000000              0.000000
                   MANLAB( AUTO)        1.000000              0.000000
                MANLAB( PLASTIC)        2.000000              0.000000
                  MANLAB( ELECT)       0.5000000              0.000000
                     CAP( STEEL)        999999.0              0.000000
                      CAP( AUTO)        650000.0              0.000000
                   CAP( PLASTIC)        60000.00              0.000000
                     CAP( ELECT)        999999.0              0.000000
          USERATE( STEEL, STEEL)       -1.000000              0.000000
           USERATE( STEEL, AUTO)       0.2000000E-01          0.000000
        USERATE( STEEL, PLASTIC)       0.1000000E-01          0.000000
          USERATE( STEEL, ELECT)        0.000000              0.000000
           USERATE( AUTO, STEEL)       0.8000000              0.000000
            USERATE( AUTO, AUTO)       -1.000000              0.000000
         USERATE( AUTO, PLASTIC)       0.1100000              0.000000
           USERATE( AUTO, ELECT)       0.1500000              0.000000
        USERATE( PLASTIC, STEEL)       0.2000000              0.000000
         USERATE( PLASTIC, AUTO)       0.3000000E-01          0.000000
      USERATE( PLASTIC, PLASTIC)       -1.000000              0.000000
        USERATE( PLASTIC, ELECT)       0.5000000E-01          0.000000
          USERATE( ELECT, STEEL)       0.1000000E-01          0.000000
           USERATE( ELECT, AUTO)       0.1000000E-01          0.000000
        USERATE( ELECT, PLASTIC)       0.5000000E-01          0.000000
          USERATE( ELECT, ELECT)       -1.000000              0.000000

                             Row    Slack or Surplus      Dual Price
                          PROFIT       0.4354312E+09        1.000000
                  NETUSE( STEEL)        0.000000            500.0000
                  CAPLIM( STEEL)        606040.7            0.000000
                   NETUSE( AUTO)        0.000000            1500.000
                   CAPLIM( AUTO)        174166.7            0.000000
                NETUSE( PLASTIC)        0.000000            3296.875
                CAPLIM( PLASTIC)        0.000000            2082.656
                  NETUSE( ELECT)        0.000000            421.8750
                  CAPLIM( ELECT)        925624.0            0.000000
                          MANLIM        0.000000            374.0625
```

The solution indicates the best way of selling Islandia's steel, automotive, electronics, plastics, and manpower resources is in the form of automobiles.

This problem would fit the classical input-output model format of Leontief if, instead of maximizing profits, target levels were set for the export (or consumption) of steel, automotive, and plastics. The problem would then be to determine the production levels necessary to support the specified export/consumption levels. In this case, the objective function is irrelevant.

A natural generalization is to allow alternative technologies for producing various commodities. These various technologies may correspond to the degree of mechanization or the form of energy consumed (e.g., gas, coal, or hydroelectric).

8.9.7 Activity/Resource Diagrams

The graphical approach for depicting a model can be extended to arbitrary LP models. The price one must pay to represent a general LP graphically is one must introduce an additional component type into the network. There are two component types in such a diagram: (1) activities, which correspond to variables and are denoted by a square, and (2) resources, which correspond to constraints and are denoted by a circle. Each constraint can be thought of as corresponding to some commodity and, in words, as saying "uses of commodity ≤ sources of commodity". The arrows incident to a square correspond to the resources, commodities, or constraints with which that variable has an interaction. The arrows incident to a circle must obviously then correspond to the activities or decision variables with which the constraint has an interaction.

Example: The Vertically Integrated Farmer

A farmer has 120 acres that can be used for growing wheat or corn. The yield is 55 bushels of wheat or 95 bushels of corn per acre per year. Any fraction of the 120 acres can be devoted to growing wheat or corn. Labor requirements are 4 hours per acre per year, plus 0.15 hours per bushel of wheat, and 0.70 hours per bushel of corn. Cost of seed, fertilizer, etc., is 20 cents per bushel of wheat produced and 12 cents per bushel of corn produced. Wheat can be sold for $1.75 per bushel and corn for $0.95 per bushel. Wheat can be bought for $2.50 per bushel and corn for $1.50 per bushel.

In addition, the farmer may raise pigs and/or poultry. The farmer sells the pigs or poultry when they reach the age of one year. A pig sells for $40. He measures the poultry in terms of coops. One coop brings in $40 at the time of sale. One pig requires 25 bushels of wheat or 20 bushels of corn, plus 25 hours of labor and 25 square feet of floor space. One coop of poultry requires 25 bushels of corn or 10 bushels of wheat, plus 40 hours of labor and 15 square feet of floor space.

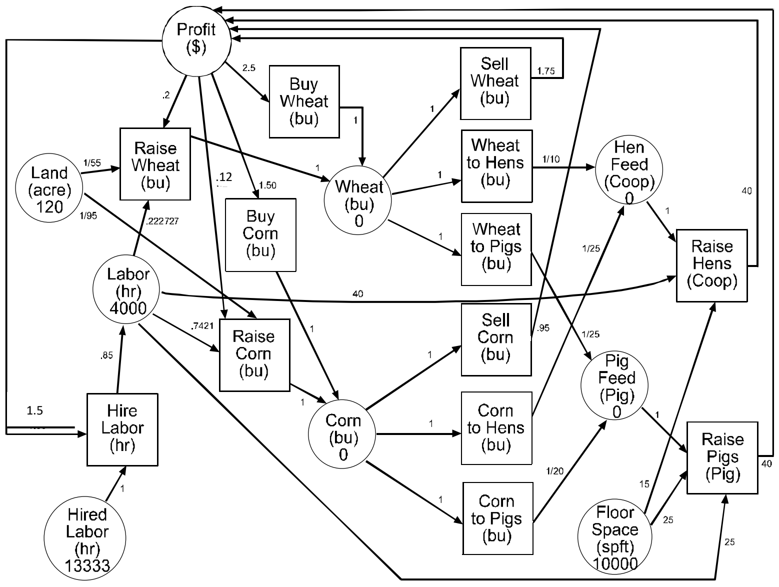
The farmer has 10,000 square feet of floor space. He has available per year 2,000 hours of his own time and another 2,000 hours from his family. He can hire labor at $1.50 per hour. However, for each hour of hired labor, 0.15 hour of the farmer's time is required for supervision. How much land should be devoted to corn and to wheat, and how many pigs and/or poultry should be raised to maximize the farmer's profits? This problem is based on an example in chapter 12 of Hadley (1962).

You may find it convenient to use the following variables for this problem:

WR	Wheat raised (in bushels)
CR	Corn harvested (in bushels)
PS	Pigs raised and sold
HS	Hens raised and sold (number of coops)
LB	Labor hired (in hours)
WS	Wheat marketed or sold (in bushels)
CS	Corn marketed or sold (in bushels)
CH	Corn used to feed hens (in bushels)
WH	Wheat used to feed hens (in bushels)
CP	Corn used to feed pigs (in bushels)
WP	Wheat used to feed pigs (in bushels)
CB	Corn bought (in bushels)
WB	Wheat bought (in bushels)

The activity-resource diagram for the preceding problem is shown in Figure 8.10:

Figure 8.10 An Activity-Resource Diagram



Some things to note about an activity-resource diagram are:

- Each rectangle in the diagram corresponds to a decision variable in the formulation.
- Each circle in the diagram corresponds to a constraint or the objective.
- Each arrow in the diagram corresponds to a coefficient in the formulation.
- Associated with each circle or rectangle is a unit of measure (e.g., hours or bushels).
- The units or dimension of each arrow is:
  "Units of the circle" per "unit of the rectangle."

Below is the formulation corresponding to the above diagram.

```
 ! All constraints are in Uses <= Sources form;
[PROFIT] MAX= 1.75*WS +.95*CS  +40*PS +40*HS -1.5*LB -.2*WR -.12*CR -1.5*CB -2.5*WB;
[LAND]    (1/55)*WR + (1/95)*CR <= 120 ;                                  ! Acres;
[LABOR]   (.15+4/55)*WR + (.7+4/95)*CR + 40*HS + 25*PS <= .85*LB + 4000 ; ! Hours;
[HRDLABOR] LB <= 2000/.15;                   ! Hours;
[WHEAT]    WS + WH + WP <= WB + WR;          ! Bushels);
[CORN]     CS + CH + CP <= CB + CR;          ! Bushels;
[HENFEED]  HS <= (1/25)*CH + (1/10)*WH;  ! Coops;
[PIGFEED]  PS <= (1/20)*CP + (1/25)*WP;  ! Pigs;
[FLOORSP]  25*PS + 15*HS  <= 10000;      ! Square feet;
```

Notice that there is a one-to-one correspondence between the rows of the formulation and the round nodes of the diagram, and a one-to-one correspondence between the variables of the formulation and the square "hyper-arcs" of the diagram. The solution is:

```
      Variable          Value        Reduced Cost
            WS       5967.500           0.000000
            CS       0.000000           0.4122697
            PS       0.000000           0.000000
            HS       63.25000           0.000000
            LB       0.000000           1.021875
            WR       6600.000           0.000000
            CR       0.000000           0.000000
            CB       0.000000           0.1377303
            WB       0.000000           0.7500000
            WH       632.5000           0.000000
            WP       0.000000           0.7125000
            CH       0.000000           0.6622697
            CP       0.000000           0.0653947

           Row    Slack or Surplus      Dual Price
        PROFIT       11653.12             1.000000
          LAND       0.000000             78.35938
         LABOR       0.000000           0.5625000
      HRDLABOR       13333.33             0.000000
         WHEAT       0.000000             1.750000
          CORN       0.000000             1.362270
       HENFEED       0.000000             17.50000
       PIGFEED       0.000000             25.93750
       FLOORSP       9051.250             0.000000
```

Notice that the most profitable use of land is to raise wheat. The most profitable use of the farmer's own labor and floor space is to use it, plus some wheat, to raise hens.

8.9.8 Spanning Trees

Another simple yet important network-related problem is the spanning tree problem. It arises, for example, in the installation of utilities such as cable, power lines, roads, and sewers to provide services to homes in newly developed regions. Given a set of homes to be connected, we want to find a minimum cost network, so every home is connected to the network. A reasonable approximation to the cost of the network is the sum of the costs of the arcs in the network. If the arcs have positive costs, then a little reflection should convince you the minimum cost network contains no loops (i.e., for any two nodes (or homes) on the network, there is exactly one path connecting them). Such a network is called a spanning tree.

A simple algorithm is available for finding a minimal cost spanning tree, see Kruskal (1956):

```
1.  Set $Y$ = {2, 3, 4 ... $n$} (i.e., the set of nodes *yet* to be connected).
    $A$ = {1} (i.e., the set of *already* connected nodes). We may arbitrarily define node 1 as the
    root of the tree.
2.  If $Y$ is empty, then we are done,
3.  else find the shortest arc $(i,j)$ such that $i$ is in $A$ and $j$ is in $Y$.
4.  Add arc $(i, j)$ to the network and
    set $A = A + j$,
        $Y = Y - j$.
5.  Go to (2).
```

Because of the above simple and efficient algorithm, LP is not needed to solve the minimum spanning tree problem. In fact, formulating the minimum spanning tree problem as an LP is a bit tedious.

The following illustrates a LINGO model for a spanning tree. This model does not explicitly solve it as above, but just solves it as a straightforward integer program:

```
MODEL:     !  (MNSPTREE);
!Given a set of nodes and the distance between each pair, find
the shortest total distance of links on the network to connect
all the nodes. This is the classic minimal spanning tree (MST)
problem;
SETS:
    CITY: LVL;
        ! LVL( I) = level of city I in tree. LVL( 1) = 0;
    LINK( CITY, CITY):
         DIST,  ! The distance matrix;
          X;    ! X( I,J) = 1 if we use link I, J;
ENDSETS
 ! This model finds the minimum cost network connecting Atlanta,
  Chicago, Cincinnati, Houston, LA, and Montreal so that
  messages can be sent from Atlanta (base) to all other cities;
DATA:
 CITY= ATL  CHI  CIN  HOU  LAX  MON;
  ! Distance matrix need not be symmetric. City 1 is base;
 DIST =  0   702  454  842 2396 1196 !from Atlanta;
       702    0  324 1093 2136  764 !from Chicago;
       454  324    0 1137 2180  798 !from Cinci;
       842 1093 1137    0 1616 1857 !from Houston;
      2396 2136 2180 1616    0 2900 !from LA;
      1196  764  798 1857 2900    0;!from Montreal;
ENDDATA
!-----------------------------------------------;
!The model size: Warning, may be slow for N > 8;
N = @SIZE( CITY);
!The objective is to minimize total dist. of links;
MIN = @SUM( LINK: DIST * X);
!For city K, except the base, ... ;
@FOR( CITY( K)| K #GT# 1: ! It must be entered;
   @SUM( CITY( I)| I #NE# K: X( I, K)) = 1;
!If there is a link from J-K, then LVL(K)=LVL(J)+1.
  Note:These are not very powerful for large problems;
   @FOR( CITY( J)| J #NE# K:
      LVL( K) >= LVL( J) + X( J, K)
        - ( N - 2) * ( 1 - X( J, K))
        + ( N - 3) * X( K, J); ); );
  LVL( 1) = 0;  ! City 1 has level 0;
!There must be an arc out of city 1;
@SUM( CITY( J)| J #GT# 1: X( 1, J)) >= 1;
!Make the X's 0/1;
@FOR( LINK: @BIN( X); );
!The level of a city except the base is at least 1 but no more than N-
1, and is 1 if link to the base;
@FOR( CITY( K)| K #GT# 1:
   @BND( 1, LVL( K), 999999);
   LVL( K) <= N - 1 - ( N - 2) * X( 1, K);  );
END
```

The solution is:

```
Optimal solution found at step:        16
Objective value:                 4000.000

   Variable           Value        Reduced Cost
          N        6.000000         0.0000000
   LVL( CHI)        2.000000         0.0000000
   LVL( CIN)        1.000000         0.0000000
   LVL( HOU)        1.000000         0.0000000
   LVL( LAX)        2.000000         0.0000000
   LVL( MON)        3.000000         0.0000000
X( ATL, CIN)        1.000000          454.0000
X( ATL, HOU)        1.000000          842.0000
X( CHI, MON)        1.000000          764.0000
X( CIN, CHI)        1.000000          324.0000
X( HOU, LAX)        1.000000          1616.000
```

The solution indicates Atlanta should connect to Cincinnati and Houston. Houston, in turn connects to LA. Cincinnati connects to Chicago, and Chicago connects to Montreal.

8.9.9 Steiner Trees

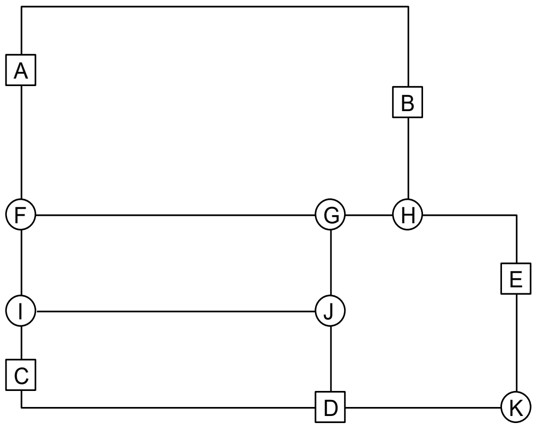
A Steiner tree is a generalization of a minimal spanning tree. The difference is, for a given network, only a specified subset of the nodes need be connected in a Steiner tree. Providing network services in a new housing development is a simple example, such as communication cable, sewer lines, water lines, and roads. Each house must be connected to the network, but not all possible nodes or junctions in the candidate network need be included.

Example

The first computer, an IBM RS6000, to beat a grandmaster, Gary Kasparov, at the game of chess, contained electronic chips designed with the help of Steiner-tree-like optimization methods. A typical VLSI (Very Large Scale Integrated) chip on this computer was less than 2 millimeters on a side. Nevertheless, it might contain over 120 meters of connecting pathways for connecting the various devices on the chip. An important part of increasing the speed of a chip is reducing the length of the paths on the chip. Figure 8.11 shows a chip on which five devices must be connected on a common tree.

Because of previous placement of various devices, the only available links are the ones indicated in the figure. The square nodes, *A*, *B*, *C*, *D*, and *E*, must be connected. The round nodes, *F*, *G*, etc., may be used, but need not be connected. What set of links should be used to minimize the total distance of links?

Figure 8.11 Steiner Tree Problem



Finding a minimal length Steiner tree is considerably harder than finding a minimal length spanning tree. For small problems, the following LINGO model will find minimal length Steiner trees. The data correspond to the network in Figure 8.11:

```
MODEL:    !  (STEINERT);
!Given a set of nodes, the distance between them, and a specified
subset of the nodes, find the set of links so that the total distance
is minimized, and there is a (unique) path between every pair of
nodes in the specified subset. This is called a Steiner tree problem;
SETS:
 ALLNODE : U;
  ! U( I) = level of node I in the tree;
                    ! U( 1) = 0;
 MUSTNOD( ALLNODE); ! The subset of nodes that must be connected;
 LINK( ALLNODE, ALLNODE):
         DIST,  ! The distance matrix;
          X;    ! X( I, J) = 1 if we use link I, J;
ENDSETS
```

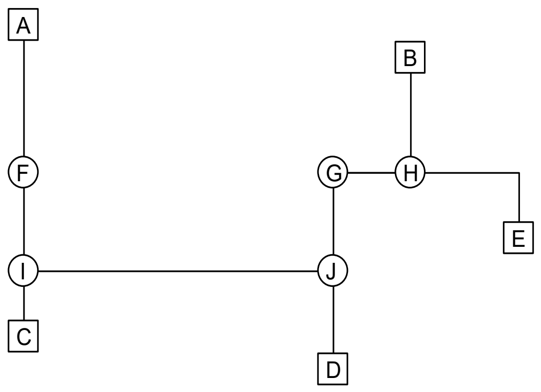
```
DATA:
 ALLNODE=  ! Distance matrix need not be symmetric;
       A   B   C   D   E   F   G   H   I   J   K;
 DIST =0  14 999 999 999   4 999 999 999 999 999
      14   0 999 999 999 999 999   3 999 999 999
     999 999   0   9 999 999 999 999   2 999 999
     999 999   9   0 999 999 999 999 999   3   6
     999 999 999 999   0 999 999   5 999 999   3
       4 999 999 999 999   0 999 999   3 999 999
     999 999 999 999 999 999   0   2 999   3 999
     999   3 999 999   5 999   2   0 999 999 999
     999 999   2 999 999   3 999 999   0   8 999
     999 999 999   3 999 999   3 999   8   0 999
     999 999 999   6   3 999 999 999 999 999   0;
! The subset of nodes that must be connected.
  The first node must be a must-do node;
MUSTNOD =  A B C D E;
ENDDATA
!-----------------------------------------------;
! The model size: Warning, may be slow for N > 8;
N = @SIZE( ALLNODE);
! Objective is minimize total distance of links;
MIN = @SUM( LINK: DIST * X);
! For each must-do node K, except the base, ... ;
@FOR( MUSTNOD( K)| K #GT# 1:
! It must be entered;
  @SUM( ALLNODE( I)| I #NE# K: X( I, K)) = 1;
! Force U(J)=number arcs between node J and node 1. Note: This is not
very strong for large problems;
@FOR( ALLNODE( J)| J #GT# 1 #AND# J #NE# K:
   U( J) >= U( K) + X ( K, J) -
    ( N - 2) * ( 1 - X( K, J)) +
    ( N - 3) * X( J, K); );
  );
! There must be an arc out of node 1;
@SUM( ALLNODE( J)| J #GT# 1: X( 1, J)) >= 1;
!If an arc out of node J, there must be an arc in;
@FOR( ALLNODE( J)| J #GT# 1:
 @FOR( ALLNODE( K)| K #NE# J:
  @SUM( ALLNODE( I)| I #NE# K #AND# I #NE# J:
                X( I, J)) >= X( J, K);
     ); );
! Make the X's 0/1;
@FOR( LINK: @BIN( X); );
! Level of a node except the base is at least 1, no more than N-1, and
is 1 if link to the base;
@FOR( ALLNODE( K)| K #GT# 1:
 @BND( 1, U( K), 999999);
 U( K) < N - 1 - ( N - 2) * X( 1, K);   );
END
```

The solution has a cost of 33. The list of links used is:

```
Optimal solution found at step:        30
Objective value:                 33.00000
Branch count:                           0
Variable           Value        Reduced Cost
X( A, F)        1.000000            4.000000
X( F, I)        1.000000            3.000000
X( G, H)        1.000000            2.000000
X( H, B)        1.000000            3.000000
X( H, E)        1.000000            5.000000
X( I, C)        1.000000            2.000000
X( I, J)        1.000000            8.000000
X( J, D)        1.000000            3.000000
X( J, G)        1.000000            3.000000
```

The corresponding minimal length Steiner tree appears in Figure 8.12:

Figure 8.12 Minimal Length Steiner Tree



Notice node K is not in the tree. This example is missing two important features of real chip design: a) It shows only two-dimensional paths. In fact, three-dimensional paths are possible (and typically needed) by adding vertical layers to the chip. b) It only shows one tree; whereas, in fact, there may be many distinct trees to be constructed (e.g., some devices need to be connected to electrical "ground", others need to be connected to the clock signal, etc.). This might be handled by solving the trees sequentially, the more complicated trees first.

8.10 Nonlinear Networks

There are a number of important problems where the constraints describe a network problem. However, either the objective is nonlinear, or there are additional conditions on the network that are nonlinear. The first example describes a transportation problem where the value of shipping or assigning an item to a destination depends upon (a) how many items have already been shipped to that destination, and (b) the type of item and type of destination. In the military, this kind of problem is known as a weapons or target assignment problem. If we define:

$x(i,j)$ = number of units of type j assigned to task i;
$p(i,j)$ = Prob{ a unit of type j will not successfully complete task i}.

Then, assuming independence, the probability task i will not be completed is:

$$p(i,1)^{x(i,1)}\, p(i,2)^{x(i,2)} \ldots p(i,n)^{x(i,n)}.$$

The log of the proability that task i will not be completed is:

$$x(i,1) * \log[p(i,1)] + x(i,2) * \log[p(i,2)] + \ldots + x(i,n) * \log[p(i,n)].$$

A reasonable objective is to maximize the expected value of successfully completed tasks. The following model illustrates this idea, using data from Bracken and McCormick (1968):

```
MODEL:
!      (TARGET)  Bracken and McCormick;
SETS:
DESTN/1..20/: VALUE, DEM, LFAILP;
```

```
SOURCE/1..5/: AVAIL;
DXS( DESTN, SOURCE): PROB, VOL;
ENDSETS
DATA:
!  Probability  that  a  unit  from  source  J  will  NOT  do  the  job  at
destination I;
PROB=
1.00        .84        .96       1.00        .92
 .95        .83        .95       1.00        .94
1.00        .85        .96       1.00        .92
1.00        .84        .96       1.00        .95
1.00        .85        .96       1.00        .95
 .85        .81        .90       1.00        .98
 .90        .81        .92       1.00        .98
 .85        .82        .91       1.00       1.00
 .80        .80        .92       1.00       1.00
1.00        .86        .95        .96        .90
1.00       1.00        .99        .91        .95
1.00        .98        .98        .92        .96
1.00       1.00        .99        .91        .91
1.00        .88        .98        .92        .98
1.00        .87        .97        .98        .99
1.00        .88        .98        .93        .99
1.00        .85        .95       1.00       1.00
 .95        .84        .92       1.00       1.00
1.00        .85        .93       1.00       1.00
1.00        .85        .92       1.00       1.00;
! Units available at each source;
AVAIL= 200       100       300       150       250;
! Min units required at each destination;
DEM=
 30   0   0   0   0 100   0   0   0  40
  0   0   0  50  70  35   0   0   0  10;
! Value of satisfying destination J;
VALUE=
 60  50  50  75  40  60  35  30  25 150
 30  45 125 200 200 130 100 100 100 150;
ENDDATA
!Max sum over I:(value of destn I)
     *Prob{success at I};
    MAX = @SUM( DESTN( I): VALUE( I) *
            ( 1 - @EXP( LFAILP( I))));
! The supply constraints;@FOR( SOURCE( J):
   @SUM( DESTN( I): VOL( I, J)) <= AVAIL( J));
@FOR( DESTN( I):
!The demand constraints;
@SUM( SOURCE( J): VOL( I, J)) > DEM( I);
!Compute log of destination I failure probability;
@FREE( LFAILP( I));
 LFAILP( I) =
 @SUM(SOURCE(J): @LOG(PROB(I,J)) * VOL(I,J));); 
END
```

Observe the model could be "simplified" slightly by using the equation computing *LFAILP* (*I*) to substitute out *LFAILP* (*I*) in the objective. The time required to solve the model would probably increase substantially, however, if this substitution were made. The reason is the number of variables appearing nonlinearly in the objective increases dramatically. A general rule for nonlinear programs is:

> If you can reduce the number of variables that appear nonlinearly in the objective or a constraint by using *linear* constraints to define intermediate variables, then it is probably worth doing.

Verify the constraint defining *LFAILP* (*I*) is linear in both *LFAILP* (*I*) and *VOL* (*I*, *J*).

Notice the solution involves fractional assignments. A simple generalization of the model is to require the *VOL* () variables to be general integers:

```
Optimal solution found at step:        152
Objective value:                  1735.570

   Variable          Value        Reduced Cost
 VOL( 1, 5)        50.81594        0.0000000
 VOL( 2, 1)        13.51739        0.0000000
 VOL( 2, 2)        1.360311        0.2176940
 VOL( 2, 5)        45.40872        0.0000000
 VOL( 3, 5)        48.62891        0.0000000
 VOL( 4, 2)        23.48955        0.0000000
 VOL( 5, 2)        20.89957        0.0000000
 VOL( 6, 1)        100.0000        0.0000000
 VOL( 7, 1)        39.10010        0.0000000
 VOL( 8, 1)        27.06643        0.0000000
 VOL( 8, 5)      0.4547474E-12     0.0000000
 VOL( 9, 1)        20.31608        0.0000000
VOL( 10, 5)        51.13144        0.0000000
VOL( 11, 4)        33.19754        0.0000000
VOL( 12, 4)        40.93452        0.0000000
VOL( 13, 5)        54.01499        0.0000000
VOL( 14, 4)        58.82350        0.0000000
VOL( 14, 5)      0.5684342E-13     0.0000000
VOL( 15, 2)        26.21095        0.0000000
VOL( 15, 3)        43.78905        0.0000000
VOL( 16, 2)        24.23657        0.2176940
VOL( 16, 4)        17.04444        0.0000000
VOL( 17, 2)        3.803054        0.0000000
VOL( 17, 3)        72.03255      -0.4182476E-05
VOL( 18, 2)      0.8881784E-15     0.1489908
VOL( 18, 3)        57.55117        0.0000000
VOL( 19, 3)        64.21183        0.0000000
VOL( 20, 3)        62.41540        0.0000000
```

8.11 Problems

1. The Slick Oil Company is preparing to make next month's pipeline shipment decisions. The Los Angeles terminal will require 200,000 barrels of oil. This oil can be supplied from either Houston or Casper, Wyoming. Houston can supply oil to L.A. at a transportation cost of $.25 per barrel. Casper can supply L.A. at a transportation cost of $.28 per barrel. The St. Louis terminal will require 120,000 barrels. St. Louis can be supplied from Houston at a cost of $.18 per barrel and from Casper at a cost of $.22 per barrel. The terminal at Freshair, Indiana requires 230,000 barrels. Oil can be shipped to Freshair from Casper at a cost of $.21 per barrel, from Houston at a cost of $.19 per barrel, and from Titusville, Pa. at a cost of $.17 per barrel. Casper will have a total of 250,000 barrels available to be shipped. Houston will have 350,000 barrels available to be shipped. Because of limited pipeline capacity, no more than 180,000 barrels can be shipped from Casper to L.A. next month and no more than 150,000 barrels from Houston to L.A. The Newark, N.J. terminal will require 190,000 barrels next month. It can be supplied only from Titusville at a cost of $.14 per barrel. The Atlanta terminal will require 150,000 barrels next month. Atlanta can be supplied from Titusville at a cost of $.16 per barrel or from Houston at a cost of $.20 per barrel. Titusville will have a total of 300,000 barrels available to be shipped.

   Formulate the problem of finding the minimum transportation cost distribution plan as a linear program.
2. Louis Szathjoseph, proprietor of the Boulangerie Restaurant, knows he will need 40, 70, and 60 tablecloths on Thursday, Friday, and Saturday, respectively, for scheduled banquets. He can rent tablecloths for three days for $2 each. A tablecloth must be laundered before it can be reused. He can have them cleaned overnight for $1.50 each. He can have them laundered by regular one-day service (e.g., one used on Thursday could be reused on Saturday) for $.80 each. There are currently 20 clean tablecloths on hand with none dirty or at the laundry. Rented tablecloths need not be cleaned before returning.
   a) What are the decision variables?
   b) Formulate the LP appropriate for minimizing the total cost of renting and laundering the tablecloths. For each day, you will probably have a constraint requiring the number of clean tablecloths available to at least equal that day's demand. For each of the first two days, you will probably want a constraint requiring the number of tablecloths sent to the laundry not to exceed those that have been made dirty. Is it a network LP?
3. The Millersburg Supply Company uses a large fleet of vehicles it leases from manufacturers. The following pattern of vehicle requirements is forecast for the next 8 months:

Month	**Jan**	**Feb**	**Mar**	**Apr**	**May**	**Jun**	**Jul**	**Aug**
Vehicles Required	430	410	440	390	425	450	465	470

   Vehicles can be leased from various manufacturers at various costs and for various lengths of time. The best plans available are: three-month lease, $1,700; four-month lease, $2,000; five-month lease, $2,600. A lease can be started in any month. On January 1, there are 200 cars on lease, all of which go off lease at the end of February.
   a) Formulate an approach for minimizing Millersburg's leasing costs over the 8 months.
   b) Show that this problem is a network problem.

4. Several years ago, a university in the Westwood section of Los Angeles introduced a bidding system for assigning professors to teach courses in its business school. The table below describes a small, slightly simplified three-professor/two-course version. For the upcoming year, each professor submits a bid for each course and places limits on how many courses he or she wants to teach in each of the school's two semesters. Each professor, however, is expected to teach four courses total per year (at most three per semester).

	Prof. X	Prof. Y	Prof. Z	Sections Needed in the Year	
				Min	Max
Fall Courses	≤ 1	≤ 3	≤ 1		
Spring Courses	≤ 3	≤ 2	≤ 3		
Course A bids	6	3	8	3	7
Course B bids	4	7	2	2	8

From the table, note that: professor *Z* strongly prefers to teach course *A*; whereas, professor *X* has a slight preference for *A*. Professor *Y* does not want to teach more than two course sections in the Spring. Over both semesters, at least three sections of Course *A* must be taught. Can you formulate this problem as a network problem?

5. *Aircraft Fuel Ferrying Problem*. Fuel cost is one of the major components of variable operating cost for an airline. Some cities collect a tax on aircraft fuel sold at their airports. Thus, the cost per liter of fuel may vary noticeably from one airport to another. A standard problem with any airliner is the determination of how much fuel to take on at each stop. Fuel consumption is minimized if just sufficient fuel is taken on at each stop to fly the plane to the next stop. This policy, however, disregards the fact that fuel prices may differ from one airport to the next. Buying all the fuel at the cheapest stop may not be the cheapest policy either. This might require carrying large fuel loads that would in turn cause large amounts of fuel to be burned in ferrying the fuel. The refueling considerations at a given stop on a route are summarized by the following three numbers: (a) the minimum amount of fuel that must be on board at takeoff to make it to the next stop, (b) the cost per liter of fuel purchased at this stop, and (c) the amount of additional fuel above the minimum that is burned per liter of fuel delivered to the next stop. These figures are given below for an airplane that starts at Dallas, goes to Atlanta, then Chicago, Des Moines, St. Louis, and back to Dallas.

	Dallas	Atlanta	Chicago	Des Moines	St. Louis
a)	3100	2700	1330	1350	2500
b)	.29	.34	.35	.31	.33
c)	.04	.03	.02	.01	.04

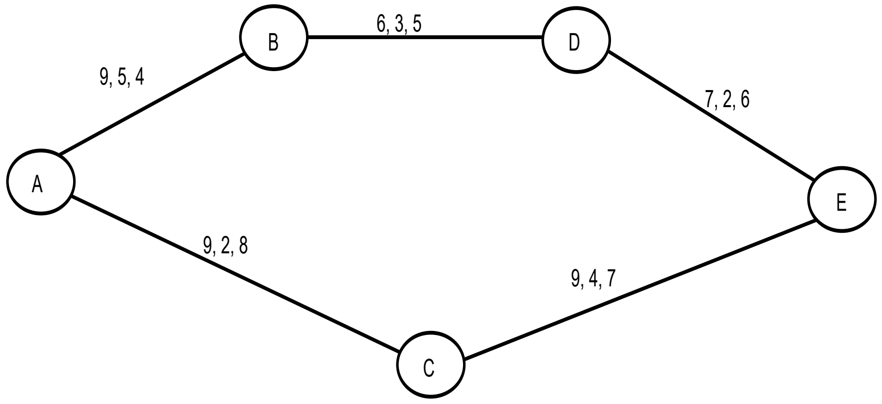
For example to fly from Dallas to Atlanta, the plane must take off with at least 3100 liters of fuel. Any fuel purchased in Dallas costs \$0.29 per liter. For each liter of fuel delivered to Atlanta (i.e., still in the tank), an additional .04 liters had to be put in at Dallas. Alternatively, each additional 1.04 liters loaded at Dallas, results in an additional liter delivered to Atlanta. The plane has a maximum fuel carrying capacity of 6000 liters, which we will assume is independent of airport. Also, assume the minimum amount of reserve fuel that must be on board for safety reasons is fixed independent of airport, so we can act as if no reserve is required.

Formulate and solve a model for deciding how much fuel to buy at each airport. Is this problem any form of a network LP?

6. Show that any LP can be converted to an equivalent LP in which every column (variable) has at most three nonzero constraint coefficients. What does this suggest about the fundamental complexity of a network LP vs. a general LP?

7. Figure 8.12 is the activity-on-arc diagram showing the precedence relations among the five activities involved in repairing a refinery. The three numbers above each arc represent (from left to right, respectively) the normal time for performing the activity in days, the time to perform the activity if crashed to the maximum extent, and the additional cost in \$1000s for each day the activity is shortened. An activity can be partially crashed. It is desired the project be completed in 15 days. Write an LP formulation for determining how much each activity should be crashed.

Figure 8.12 PERT Diagram with Crashing Allowed



8. Given n currencies, the one-period currency exchange problem is characterized by a beginning inventory vector, an exchange rate matrix, and an ending inventory requirement vector defined as follows:

n_i = amount of cash on hand in currency i, at the beginning of the period measured in units of currency i, for $i = 1, 2, ..., n$;

r_{ij} = units of currency j obtainable per unit of currency i for $i = 1, 2, ..., n$, $j = 1, 2, ..., n$. Note that $r_{ii} = 1$ and, in general, we can expect $r_{ij} < 1/r_{ji}$, for $i \neq j$.

e_i = minimum ending inventory requirement for currency i, for $i = 1, 2, ..., n$. That is, at the end of the period, we must have at least e_i units of currency i on hand.

The decision variables are:

X_{ij} = amount of currency i converted into currency j, for $i = 1, 2, ..., n$; $j = 1, 2, ..., n$.

Formulate a model for determining an efficient set of values for the X_{ij}. The formulation should have the following features:

a) If there is a "money pump" kind of arbitrage opportunity, the model will find it.

b) It should not be biased against any particular currency (i.e., the solution should be independent of which currency is called 1).

c) If a currency is worthless, you should buy no more of it than sufficient to meet the minimum requirement. A currency i is worthless if $r_{ij} = 0$, for all $j \neq i$.

9. The following linear program happens to be a network LP. Draw the corresponding network. Label the nodes and links.

```
MIN = 4 * T + 2 * U + 3 * V + 5 * W  + 6 * X + 7 * Y + 9 * Z;
   [A] T + Y + Z >= 4;
   [B] U - W - X - Z = 0;
   [C] - T + W = 1;
   [D] V + X - Y = 2;
   [E] U + V <= 7;
END
```

10. Consider a set of three flights provided by an airline to serve four cities, A, B, C, and H. The airline uses a two-fare pricing structure. The decision of how many seats or capacity to allocate to each price class is sometimes called yield or revenue management. We would like to decide upon how many seats to allocate to each fare class on each flight. Node H is a hub for changing planes. The three flights are: from A to H, H to B, and H to C. The respective flight capacities are 120, 100, and 110. Customer demand has the following characteristics:

Itinerary	Class 1 Demand	At a Price of	Class 2 Demand	At a Price of
AH	33	190	56	90
AB (via H)	24	244	43	193
AC (via H)	12	261	67	199
HB	44	140	69	80
HC	16	186	17	103

How many seats should be allocated to each class on each of the three flights? An obvious solution, if it is feasible, is to set aside enough class 1 seats on every flight, so all class 1 travelers can be accommodated. Thus, the leg AH would get 33 + 24 + 12 = 69 class 1 seats, leg HB would get 24 + 44 = 68 class 1 seats and leg HC would get 12 + 16 = 28 class 1 seats. The total revenue of this solution is $38,854. Is this the most profitable solution?

11. A common distribution system structure in many parts of the world is the three-level system composed of plants, distribution centers (DC), and outlets. A cost minimization model for a system composed of two plants (*A* & *B*), three DC's (*X*, *Y*, and *Z*), and four outlets (1, 2, 3, and 4) is shown below:

```
MIN = AX + 2 * AY + 3 * BX + BY + 2 * BZ + 5 * X1 +
      7 * X2 + 9 * Y1 + 6 * Y2 + 7 * Y3 + 8 * Z2 + 7
      * Z3 + 4 * Z4;
  AX + AY = 9;
  BX + BY + BZ = 8;
- AX - BX + X1 + X2 = 0;
- AY - BY + Y1 + Y2 + Y3 = 0;
- BZ + Z2 + Z3 + Z4 = 0;
- X1 - Y1 = - 3;
- X2 - Y2 - Z2 = - 5;
- Y3 - Z3 = - 4;
- Z4 = - 5;
END
```

Part of the solution is shown below:

```
Objective value:                  121.0000
Variable             Value          Reduced Cost
      AX          3.000000             0.0000000
      AY          6.000000             0.0000000
      BX         0.0000000              3.000000
      BY          3.000000             0.0000000
      BZ          5.000000             0.0000000
      X1          3.000000             0.0000000
      X2         0.0000000             0.0000000
      Y1         0.0000000              5.000000
      Y2          5.000000             0.0000000
      Y3          4.000000             0.0000000
      Z2         0.0000000              3.000000
      Z3         0.0000000              1.000000
      Z4          5.000000             0.0000000
```

a) Is there an alternate optimal solution to this distribution problem?

b) A trucking firm that offers services from city *Y* to city 1 would like to get more of your business. At what price per unit might you be willing to give them more business according to the above solution?

c) The demand at city 2 has been decreased to 3 units. Show how the model is changed.

d) The capacity of plant *B* has been increased to 13 units. Show how the model is changed.

e) Distribution center *Y* is actually in a large city where there is an untapped demand of 3 units that could be served directly from the DC at *Y*. Show how to include this additional demand at *Y*.

12. Labor on the first shift of a day (8 a.m. to 4 p.m.) costs \$15 per person × hour. Labor on the second (4 p.m. to midnight) and third (midnight to 8 a.m.) shifts cost \$20 per person × hour and \$25 per person × hour, respectively. A certain task requires 18 days if done with just first shift labor and costs \$8640. Second and third shift labor has the same efficiency as first shift labor. The only way of accelerating or crashing the task is to add additional shifts for one or more additional days. The total cost of the task consists solely of labor costs.

    Complete the following crash cost table for this task.

Task time in whole days	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4
Total cost	8640														

13. You are on a camping trip and wish to prepare a recipe for a certain food delight that calls for 4 cups of water. The only containers in your possession are two ungraduated steel vessels, one of 3-cup capacity, the other of 5-cup capacity. Show how you can solve this problem by drawing a certain two-dimensional network, where each node represents a specific combination of contents in your two containers.

14. Following is part of the schedule for an airline:

		City		Time		Difference
	Flight	**Depart**	**Arrive**	**Depart**	**Arrive**	**in Profit**
1	221	ORD	DEN	0800	0934	+\$3000
2	223	ORD	DEN	0900	1039	–\$4000
3	274	LAX	DEN	0800	1116	–\$3000
4	105	ORD	LAX	1100	1314	+\$10000
5	228	DEN	ORD	1100	1423	–\$2000
6	230	DEN	ORD	1200	1521	–\$3000
7	259	ORD	LAX	1400	1609	+\$4000
8	293	DEN	LAX	1400	1510	+\$1000
9	412	LAX	ORD	1400	1959	+\$7000
10	766	LAX	DEN	1600	1912	+\$2000
11	238	DEN	ORD	1800	2121	–\$4000

The airline currently flies the above schedule using standard Boeing 737 aircraft. Boeing is trying to convince the airline to use a new aircraft, the 737-XX, known affectionately as the Dos Equis. The 737-XX consumes more fuel per kilometer. However, it is sufficiently larger such that, if it carries enough passengers, it is more efficient per passenger kilometer. The "Difference in Profit" column above shows the relative profitability of using the 737-XX instead of the standard 737 on each flight. The airline is considering using at most one 737-XX.

Based on the available information, analyze the wisdom of using the 737-XX in place of one of the standard 737's.

15. The following linear program happens to be a network LP:

```
MIN = 9 * S + 4 * T + 2 * U + 3 * V + 5 * W + 6 * X + 7 * Y;
   [A] - T + W = 1;
   [B] S + T + Y  >= 4;
   [C] U - W - X - S = 0;
   [D] U + V <= 7;
   [E] V + X - Y = 2;
END
```

a) Draw the corresponding network.

b) Label the nodes and links.

16. A small, but growing, long-distance phone company, SBG Inc., is trying to decide in which markets it should try to expand. It has used the following model to decide how to maximize the calls it carries per hour:

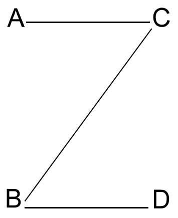
```
MAX = P1SEADNV + P2SEADNV + P1SEACHI + P2SEACHI
      + P1SEAATL + P2SEAATL + P1SEAMIA + P2SEAMIA
      + P1DNVCHI + P2DNVCHI + P1DNVATL + P2DNVATL
      + P1DNVMIA + P2DNVMIA + P1CHIATL + P2CHIATL
      + P1CHIMIA + P2CHIMIA + P1ATLMIA;
 [LSEADNV] P1SEADNV + P2SEACHI + P2SEAATL + P2SEAMIA
   + P1DNVCHI + P2DNVATL + P2DNVMIA + P2CHIATL +
   P2CHIMIA <= 95;
 [LSEACHI] P2SEADNV + P1SEACHI + P1SEAATL + P1SEAMIA
   + P1DNVCHI + P2DNVATL + P2DNVMIA + P2CHIATL +
   P2CHIMIA <= 80;
 [LDNVATL] P2SEADNV + P2SEACHI + P2SEAATL + P2SEAMIA
   + P2DNVCHI + P1DNVATL + P1DNVMIA + P2CHIATL +
   P2CHIMIA <= 200;
 [LCHIATL] P2SEADNV + P2SEACHI + P1SEAATL + P1SEAMIA
   + P2DNVCHI + P2DNVALT + P2DNVMIA + P1CHIATL +
   P1CHIMIA <= 110;
 [LATLMIA] P1SEAMIA + P2SEAMIA + P1DNVMIA + P2DNVMIA
   + P1CHIMIA + P2CHIMIA + P1ATLMIA <= 105;
 [DSEADNV] P1SEADNV + P2SEADNV <= 10;
 [DSEACHI] P1SEACHI + P2SEACHI <= 20;
 [DSEAATL] P1SEAATL + P2SEAATL <= 38;
 [DSEAMIA] P1SEAMIA + P2SEAMIA <= 33;
 [DDNVCHI] P1DNVCHI + P2DNVCHI <= 42;
 [DDNVATL] P1DNVATL + P2DNVATL <= 48;
 [DDNVMIA] P1DNVMIA + P2DNVMIA <= 23;
 [DCHIATL] P1CHIATL + P2CHIATL <= 90;
 [DCHIMIA] P1CHIMIA + P2CHIMIA <= 36;
 [DATLMIA] P1ATLMIA            <= 26;
```

Now, it would like to refine the model, so it takes into account not only revenue per call, but also modest variable costs associated with each link for carrying a call. The variable cost per typical call according to link used is shown in the table below:

	Variable Cost/Call			
	DNV	**CHI**	**ATL**	**MIA**
SEA	.11	.16	X	X
DNV		X	.15	X
CHI			.06	X
ATL				.07

An X means there is no direct link between the two cities. SBG would like to find the combination of calls to accept to maximize profit contribution. Suppose the typical revenue per call between *ATL* and *SEA* is \$1.20. Show how to modify the model just to represent the revenue and cost information for the demand between *SEA* and *ATL*.

17. Below is a four-activity project network presented in activity-on-node form, along with information on crashing opportunities for each activity:



Activity	Normal Time (days)	Crash Cost Per Day	Minimum Possible Time (days)
A	8	3	4
B	7	4	5
C	6	6	3
D	9	2	5

Complete the following tabulation of crashing cost vs. project length:

Step	Project Length	Incremental Crashing Cost/Day	Total Crashing Cost	Activities to Crash
0	16	0	0	—
1				
2				
3				
4				