3

Transportation and Assignment Models

The linear programs in Chapters 1 and 2 are all examples of classical ''activity'' models. In such models the variables and constraints deal with distinctly different kinds of activities — tons of steel produced versus hours of mill time used, or packages of food bought versus percentages of nutrients supplied. To use these models you must supply coefficients like tons per hour or percentages per package that convert a unit of activity in the variables to the corresponding amount of activity in the constraints.

This chapter addresses a significantly different but equally common kind of model, in which something is shipped or assigned, but not converted. The resulting constraints, which reflect both limitations on availability and requirements for delivery, have an especially simple form.

We begin by describing the so-called transportation problem, in which a single good is to be shipped from several origins to several destinations at minimum overall cost. This problem gives rise to the simplest kind of linear program for minimum-cost flows. We then generalize to a transportation model, an essential step if we are to manage all the data, variables and constraints effectively.

As with the diet model, the power of the transportation model lies in its adaptability. We continue by considering some other interpretations of the ''flow'' from origins to destinations, and work through one particular interpretation in which the variables represent assignments rather than shipments.

The transportation model is only the most elementary kind of minimum-cost flow model. More general models are often best expressed as networks, in which nodes — some of which may be origins or destinations — are connected by arcs that carry flows of some kind. AMPL offers convenient features for describing network flow models, including `node` and `arc` declarations that specify network structure directly. Network models and the relevant AMPL features are the topic of Chapter 15.

3.1 A linear program for the transportation problem

Suppose that we have decided (perhaps by the methods described in Chapter 1) to produce steel coils at three mill locations, in the following amounts:

GARY	Gary, Indiana	1400
CLEV	Cleveland, Ohio	2600
PITT	Pittsburgh, Pennsylvania	2900

The total of 6,900 tons must be shipped in various amounts to meet orders at seven locations of automobile factories:

FRA	Framingham, Massachusetts	900
DET	Detroit, Michigan	1200
LAN	Lansing, Michigan	600
WIN	Windsor, Ontario	400
STL	St. Louis, Missouri	1700
FRE	Fremont, California	1100
LAF	Lafayette, Indiana	1000

We now have an optimization problem: What is the least expensive plan for shipping the coils from mills to plants?

To answer the question, we need to compile a table of shipping costs per ton:

	GARY	CLEV	PITT
FRA	39	27	24
DET	14	9	14
LAN	11	12	17
WIN	14	9	13
STL	16	26	28
FRE	82	95	99
LAF	8	17	20

Let GARY:FRA be the number of tons to be shipped from GARY to FRA, and similarly for the other city pairs. Then the objective can be written as follows:

> Minimize
>> 39 GARY:FRA + 27 CLEV:FRA + 24 PITT:FRA +
>> 14 GARY:DET + 9 CLEV:DET + 14 PITT:DET +
>> 11 GARY:LAN + 12 CLEV:LAN + 17 PITT:LAN +
>> 14 GARY:WIN + 9 CLEV:WIN + 13 PITT:WIN +
>> 16 GARY:STL + 26 CLEV:STL + 28 PITT:STL +
>> 82 GARY:FRE + 95 CLEV:FRE + 99 PITT:FRE +
>> 8 GARY:LAF + 17 CLEV:LAF + 20 PITT:LAF

There are 21 decision variables in all. Even a small transportation problem like this one has a lot of variables, because there is one for each combination of mill and factory.

By supplying each factory from the mill that can ship most cheaply to it, we could achieve the lowest conceivable shipping cost. But we would then be shipping 900 tons from PITT, 1600 from CLEV, and all the rest from GARY — amounts quite inconsistent with the production levels previously decided upon. We need to add a constraint that the sum of the shipments from GARY to the seven factories is equal to the production level of 1400:

> GARY:FRA + GARY:DET + GARY:LAN + GARY:WIN +
>   GARY:STL + GARY:FRE + GARY:LAF = 1400

There are analogous constraints for the other two mills:

> CLEV:FRA + CLEV:DET + CLEV:LAN + CLEV:WIN +
>   CLEV:STL + CLEV:FRE + CLEV:LAF = 2600
>
> PITT:FRA + PITT:DET + PITT:LAN + PITT:WIN +
>   PITT:STL + PITT:FRE + PITT:LAF = 2900

There also have to be constraints like these at the factories, to ensure that the amounts shipped equal the amounts ordered. At FRA, the sum of the shipments received from the three mills must equal the 900 tons ordered:

> GARY:FRA + CLEV:FRA + PITT:FRA = 900

And similarly for the other six factories:

> GARY:DET + CLEV:DET + PITT:DET = 1200
> GARY:LAN + CLEV:LAN + PITT:LAN = 600
> GARY:WIN + CLEV:WIN + PITT:WIN = 400
> GARY:STL + CLEV:STL + PITT:STL = 1700
> GARY:FRE + CLEV:FRE + PITT:FRE = 1100
> GARY:LAF + CLEV:LAF + PITT:LAF = 1000

We have ten constraints in all, one for each mill and one for each factory. If we add the requirement that all variables be nonnegative, we have a complete linear program for the transportation problem.

We won't even try showing what it would be like to type all of these constraints into an AMPL model file. Clearly we want to set up a general model to deal with this problem.

3.2 An AMPL model for the transportation problem

Two fundamental sets of objects underlie the transportation problem: the sources or origins (mills, in our example) and the destinations (factories). Thus we begin the AMPL model with a declaration of these two sets:

```
set ORIG;
set DEST;
```

There is a supply of something at each origin (tons of steel coils produced, in our case), and a demand for the same thing at each destination (tons of coils ordered). AMPL defines nonnegative quantities like these with `param` statements indexed over a set; in this case we add one extra refinement, a `check` statement to test the data for validity:

```
param supply {ORIG} >= 0;
param demand {DEST} >= 0;

check: sum {i in ORIG} supply[i] = sum {j in DEST} demand[j];
```

The `check` statement says that the sum of the supplies has to equal the sum of the demands. The way that our model is to be set up, there can't possibly be any solutions unless this condition is satisfied. By putting it in a `check` statement, we tell AMPL to test this condition after reading the data, and to issue an error message if it is violated.

For each combination of an origin and a destination, there is a transportation cost and a variable representing the amount transported. Again, the ideas from previous chapters are easily adapted to produce the appropriate AMPL statements:

```
param cost {ORIG,DEST} >= 0;
var Trans {ORIG,DEST} >= 0;
```

For a particular origin `i` and destination `j`, we ship `Trans[i,j]` units from `i` to `j`, at a cost of `cost[i,j]` per unit; the total cost for this pair is

```
cost[i,j] * Trans[i,j]
```

Adding over all pairs, we have the objective function:

```
minimize Total_Cost:
    sum {i in ORIG, j in DEST} cost[i,j] * Trans[i,j];
```

which could also be written as

```
sum {j in DEST, i in ORIG} cost[i,j] * Trans[i,j];
```

or as

```
sum {i in ORIG} sum {j in DEST} cost[i,j] * Trans[i,j];
```

As long as you express the objective in some mathematically correct way, AMPL will sort out the terms.

It remains to specify the two collections of constraints, those at the origins and those at the destinations. If we name these collections `Supply` and `Demand`, their declarations will start as follows:

```
subject to Supply {i in ORIG}:  ...
subject to Demand {j in DEST}:  ...
```

To complete the `Supply` constraint for origin `i`, we need to say that the sum of all shipments out of `i` is equal to the supply available. Since the amount shipped out of `i` to a particular destination `j` is `Trans[i,j]`, the amount shipped to all destinations must be

```
set ORIG;    # origins
set DEST;    # destinations

param supply {ORIG} >= 0;   # amounts available at origins
param demand {DEST} >= 0;   # amounts required at destinations

   check: sum {i in ORIG} supply[i] = sum {j in DEST} demand[j];

param cost {ORIG,DEST} >= 0;   # shipment costs per unit
var Trans {ORIG,DEST} >= 0;    # units to be shipped

minimize Total_Cost:
   sum {i in ORIG, j in DEST} cost[i,j] * Trans[i,j];

subject to Supply {i in ORIG}:
   sum {j in DEST} Trans[i,j] = supply[i];

subject to Demand {j in DEST}:
   sum {i in ORIG} Trans[i,j] = demand[j];
```

Figure 3-1a: Transportation model (`transp.mod`).

```
   sum {j in DEST} Trans[i,j]
```

Since we have already defined a parameter `supply` indexed over origins, the amount available at `i` is `supply[i]`. Thus the constraint is

```
   subject to Supply {i in ORIG}:
     sum {j in DEST} Trans[i,j] = supply[i];
```

(Note that the names `supply` and `Supply` are unrelated; AMPL distinguishes upper and lower case.) The other collection of constraints is much the same, except that the roles of `i` in `ORIG`, and `j` in `DEST`, are exchanged, and the sum equals `demand[j]`.

We can now present the complete transportation model, Figure 3-1a. As you might have noticed, we have been consistent in using the index `i` to run over the set `ORIG`, and the index `j` to run over `DEST`. This is not an AMPL requirement, but such a convention makes it easier to read a model. You may name your own indices whatever you like, but keep in mind that the scope of an index — the part of the model where it has the same meaning — is to the end of the expression that defines it. Thus in the `Demand` constraint

```
   subject to Demand {j in DEST}:
     sum {i in ORIG} Trans[i,j] = demand[j];
```

the scope of `j` runs to the semicolon at the end of the declaration, while the scope of `i` extends only through the summand `Trans[i,j]`. Since `i`'s scope is inside `j`'s scope, these two indices must have different names. Also an index may not have the same name as a set or other model component. Index scopes are discussed more fully, with further examples, in Section 5.5.

Data values for the transportation model are shown in Figure 3-1b. To define `DEST` and `demand`, we have used an input format that permits a set and one or more parameters indexed over it to be specified together. The set name is surrounded by colons. (We also show some comments, which can appear among data statements just as in a model.)

```
param: ORIG:   supply :=  # defines set "ORIG" and param "supply"
         GARY   1400
         CLEV   2600
         PITT   2900 ;

param: DEST:   demand :=  # defines "DEST" and "demand"
         FRA     900
         DET    1200
         LAN     600
         WIN     400
         STL    1700
         FRE    1100
         LAF    1000 ;

param cost:
         FRA   DET   LAN   WIN   STL   FRE   LAF :=
    GARY  39    14    11    14    16    82     8
    CLEV  27     9    12     9    26    95    17
    PITT  24    14    17    13    28    99    20 ;
```

Figure 3-1b: Data for transportation model (`transp.dat`).

If the model is stored in a file `transp.mod` and the data in `transp.dat`, we can solve the linear program and examine the output:

```
ampl: model transp.mod;
ampl: data transp.dat;
ampl: solve;
CPLEX 8.0.0: optimal solution; objective 196200
12 dual simplex iterations (0 in phase I)

ampl: display Trans;
Trans [*,*] (tr)
:      CLEV   GARY   PITT      :=
DET    1200      0      0
FRA       0      0    900
FRE       0   1100      0
LAF     400    300    300
LAN     600      0      0
STL       0      0   1700
WIN     400      0      0
 ;
```

By displaying the variable `Trans`, we see that most destinations are supplied from a single mill, but CLEV, GARY and PITT all ship to LAF.

It is instructive to compare this solution to one given by another solver, SNOPT:

```
ampl: option solver snopt;
ampl: solve;
SNOPT 6.1-1: Optimal solution found.
15 iterations, objective 196200
```

```
ampl: display Trans;
Trans [*,*] (tr)
:      CLEV    GARY    PITT     :=
DET    1200       0       0
FRA       0       0     900
FRE       0    1100       0
LAF     400       0     600
LAN     600       0       0
STL       0     300    1400
WIN     400       0       0
;
```

The minimum cost is still 196200, but it is achieved in a different way. Alternative optimal solutions such as these are often exhibited by transportation problems, particularly when the coefficients in the objective function are round numbers.

Unfortunately, there is no easy way to characterize all the optimal solutions. You may be able to get a better choice of optimal solution by working with several objectives, however, as we will illustrate in Section 8.3.

3.3 Other interpretations of the transportation model

As the name suggests, a transportation model is applicable whenever some material is being shipped from a set of origins to a set of destinations. Given certain amounts available at the origins, and required at the destinations, the problem is to meet the requirements at a minimum shipping cost.

Viewed more broadly, transportation models do not have to be concerned with the shipping of ''materials''. They can be applied to the transportation of anything, provided that the quantities available and required can be measured in some units, and that the transportation cost per unit can be determined. They might be used to model the shipments of automobiles to dealers, for example, or the movement of military personnel to new assignments.

In an even broader view, transportation models need not deal with ''shipping'' at all. The quantities at the origins may be merely associated with various destinations, while the objective measures some value of the association that has nothing to do with actually moving anything. Often the result is referred to as an ''assignment'' model.

As one particularly well-known example, consider a department that needs to assign some number of people to an equal number of offices. The origins now represent individual people, and the destinations represent individual offices. Since each person is assigned one office, and each office is occupied by one person, all of the parameter values `supply[i]` and `demand[j]` are 1. We interpret `Trans[i,j]` as the ''amount'' of person i that is assigned to office j; that is, if `Trans[i,j]` is 1 then person i will occupy office j, while if `Trans[i,j]` is 0 then person i will not occupy office j.

What of the objective? One possibility is to ask people to rank the offices, giving their first choice, second choice, and so forth. Then we can let `cost[i,j]` be the rank

```
set ORIG := Coullard Daskin Hazen Hopp Iravani Linetsky
            Mehrotra Nelson Smilowitz Tamhane White ;

set DEST := C118 C138 C140 C246 C250 C251 D237 D239 D241 M233 M239;

param supply default 1 ;

param demand default 1 ;

param cost:
          C118 C138 C140 C246 C250 C251 D237 D239 D241 M233 M239 :=
  Coullard   6    9    8    7   11   10    4    5    3    2    1
  Daskin    11    8    7    6    9   10    1    5    4    2    3
  Hazen      9   10   11    1    5    6    2    7    8    3    4
  Hopp      11    9    8   10    6    5    1    7    4    2    3
  Iravani    3    2    8    9   10   11    1    5    4    6    7
  Linetsky  11    9   10    5    3    4    6    7    8    1    2
  Mehrotra   6   11   10    9    8    7    1    2    5    4    3
  Nelson    11    5    4    6    7    8    1    9   10    2    3
  Smilowitz 11    9   10    8    6    5    7    3    4    1    2
  Tamhane    5    6    9    8    4    3    7   10   11    2    1
  White     11    9    8    4    6    5    3   10    7    2    1 ;
```

Figure 3-2: Data for assignment problem (`assign.dat`).

that person `i` gives to office `j`. This convention lets each objective function term `cost[i,j] * Trans[i,j]` represent the preference of person `i` for office `j`, if person `i` is assigned to office `j` (`Trans[i,j]` equals 1), or zero if person `i` is not assigned to office `j` (`Trans[i,j]` equals 0). Since the objective is the sum of all these terms, it must equal the sum of all the nonzero terms, which is the sum of everyone's rankings for the offices to which they were assigned. By minimizing this sum, we can hope to find an assignment that will please a lot of people.

To use the transportation model for this purpose, we need only supply the appropriate data. Figure 3-2 is one example, with 11 people to be assigned to 11 offices. The `default` option has been used to set all the `supply` and `demand` values to 1 without typing all the 1's. If we store this data set in `assign.dat`, we can use it with the transportation model that we already have:

```
ampl: model transp.mod;
ampl: data assign.dat;
ampl: solve;
CPLEX 8.0.0: optimal solution; objective 28
24 dual simplex iterations (0 in phase I)
```

By setting the option `omit_zero_rows` to 1, we can print just the nonzero terms in the objective. (Options for displaying results are presented in Chapter 12.) This listing tells us each person's assigned room and his or her preference for it:

```
ampl: option omit_zero_rows 1;
ampl: display {i in ORIG, j in DEST} cost[i,j] * Trans[i,j];
cost[i,j]*Trans[i,j] :=
Coullard  C118   6
Daskin    D241   4
Hazen     C246   1
Hopp      D237   1
Iravani   C138   2
Linetsky  C250   3
Mehrotra  D239   2
Nelson    C140   4
Smilowitz M233   1
Tamhane   C251   3
White     M239   1
;
```

The solution is reasonably successful, although it does assign two fourth choices and one sixth choice.

It is not hard to see that when all the `supply[i]` and `demand[j]` values are 1, any `Trans[i,j]` satisfying all the constraints must be between 0 and 1. But how did we know that every `Trans[i,j]` would equal either 0 or 1 in the optimal solution, rather than, say, ½? We were able to rely on a special property of transportation models, which guarantees that as long as all supply and demand values are integers, and all lower and upper bounds on the variables are integers, there will be an optimal solution that is entirely integral. Moreover, we used a solver that always finds one of these integral solutions. But don't let this favorable result mislead you into assuming that integrality can be assured in all other circumstances; even in examples that seem to be much like the transportation model, finding integral solutions can require a special solver, and a lot more work. Chapter 20 discusses issues of integrality at length.

A problem of assigning 100 people to 100 rooms has ten thousand variables; assigning 1000 people to 1000 rooms yields a million variables. In applications on this scale, however, most of the assignments can be ruled out in advance, so that the number of actual decision variables is not too large. After looking at an initial solution, you may want to rule out some more assignments — in our example, perhaps no assignment to lower than fifth choice should be allowed — or you may want to force some assignments to be made a certain way, in order to see how the rest could be done optimally. These situations require models that can deal with subsets of pairs (of people and offices, or origins and destinations) in a direct way. AMPL's features for describing pairs and other ''compound'' objects are the subject of Chapter 6.

Exercises

3-1. This transportation model, which deals with finding a least cost shipping schedule, comes from Dantzig's *Linear Programming and Extensions.* A company has plants in Seattle and San Diego, with capacities 350 and 600 cases per week respectively. It has customers in New York, Chicago, and Topeka, which order 325, 300, and 275 cases per week. The distances involved are:

	New York	Chicago	Topeka
Seattle	2500	1700	1800
San Diego	2500	1800	1400

The shipping cost is $90 per case per thousand miles. Formulate this model in AMPL and solve it to determine the minimum cost and the amounts to be shipped.

3-2. A small manufacturing operation produces six kinds of parts, using three machines. For the coming month, a certain number of each part is needed, and a certain number of parts can be accommodated on each machine; to complicate matters, it does not cost the same amount to make the same part on different machines. Specifically, the costs and related values are as follows:

			Part				
Machine	1	2	3	4	5	6	Capacity
1	3	3	2	5	2	1	80
2	4	1	1	2	2	1	30
3	2	2	5	1	1	2	160
Required	10	40	60	20	20	30	

(a) Using the model in Figure 3-1a, create a file of data statements for this problem; treat the machines as the origins, and the parts as the destinations. How many of each part should be produced on each machine, so as to minimize total cost?

(b) If the capacity of machine 2 is increased to 50, the manufacturer may be able to reduce the total cost of production somewhat. What small change to the model is necessary to analyze this situation? How much is the total cost reduced, and in what respects does the production plan change?

(c) Now suppose that the capacities are given in hours, rather than in numbers of parts, and that it takes a somewhat different number of hours to make the same part on different machines:

			Part				
Machine	1	2	3	4	5	6	Capacity
1	1.3	1.3	1.2	1.5	1.2	1.1	50
2	1.4	1.1	1.1	1.2	1.2	1.1	90
3	1.2	1.2	1.5	1.1	1.1	1.2	175

Modify the supply constraint so that it limits total time of production at each ''origin'' rather than the total quantity of production. How is the new optimal solution different? On which machines is all available time used?

(d) Solve the preceding problem again, but with the objective function changed to minimize total machine-hours rather than total cost.

3-3. This exercise deals with generalizations of the transportation model and data of Figure 3-1.

(a) Add two parameters, `supply_pct` and `demand_pct`, to represent the maximum fraction of a mill's supply that may be sent to any one factory, and the maximum fraction of a factory's

demand that may be satisfied by any one mill. Incorporate these parameters into the model of Figure 3-1a.

Solve for the case in which no more than 50% of a mill's supply may be sent to any one factory, and no more than 85% of a factory's demand may be satisfied by any one mill. How does this change the minimum cost and the optimal amounts shipped?

(b) Suppose that the rolling mills do not produce their own slabs, but instead obtain slabs from two other plants, where the following numbers of tons are to be made available:

```
    MIDTWN  2700
    HAMLTN  4200
```

The cost per ton of shipping a slab from a plant to a mill is as follows:

```
            GARY    CLEV    PITT
    MIDTWN   12       8      17
    HAMLTN   10       5      13
```

All other data values are the same as before, but with `supply_pct` reinterpreted as the maximum fraction of a plant's supply that may be sent to any one mill.

Formulate this situation as an AMPL model. You will need two indexed collections of variables, one for the shipments from plants to mills, and one for the shipments from mills to factories. Shipments from each mill will have to equal supply, and shipments to each factory will have to equal demand as before; also, shipments out of each mill will have to equal shipments in.

Solve the resulting linear program. What are the shipment amounts in the minimum-cost solution?

(c) In addition to the differences in shipping costs, there may be different costs of production at the plants and mills. Explain how production costs could be incorporated into the model.

(d) When slabs are rolled, some fraction of the steel is lost as scrap. Assuming that this fraction may be different at each mill, revise the model to take scrap loss into account.

(e) In reality, scrap is not really lost, but is sold for recycling. Make a further change to the model to account for the value of the scrap produced at each mill.

3-4. This exercise considers variations on the assignment problem introduced in Section 3.3.

(a) Try reordering the list of members of DEST in the data (Figure 3-2), and solving again. Find a reordering that causes your solver to report a different optimal assignment.

(b) An assignment that gives even one person a very low-ranked office may be unacceptable, even if the total of the rankings is optimized. In particular, our solution gives one individual her sixth choice; to rule this out, change all preferences of six or larger in the cost data to 99, so that they will become very unattractive. (You'll learn more convenient features for doing the same thing in later chapters, but this crude approach will work for now.) Solve the assignment problem again, and verify that the result is an equally good assignment in which no one gets worse than fifth choice.

Now apply the same approach to try to give everyone no worse than fourth choice. What do you find?

(c) Suppose now that offices C118, C250 and C251 become unavailable, and you have to put two people each into C138, C140 and C246. Add 20 to each ranking for these three offices, to reflect the fact that anyone would prefer a private office to a shared one. What other modifications to the model and data would be necessary to handle this situation? What optimal assignment do you get?

(d) Some people may have seniority that entitles them to greater consideration in their choice of office. Explain how you could enhance the model to use seniority level data for each person.