CHAPTER 23

Additional Special Types of Linear Programming Problems

Chapter 3 emphasized the wide applicability of linear programming. Chapters 8 and 9 then described some of the special types of linear programming problems that often arise, including the transportation problem (Sec. 8.1), the assignment problem (Sec. 8.3), the shortest-path problem (Sec. 9.3), the maximum flow problem (Sec. 9.5), and the minimum cost flow problem (Sec. 9.6). These latter chapters also presented streamlined versions of the simplex method for solving these problems very efficiently.

We continue to broaden our horizons in this chapter by discussing some additional special types of linear programming problems. These additional types often share several key characteristics in common with the special types presented in Chapters 8 and 9. The first is that they all arise frequently in a variety of contexts. They also tend to require a very large number of constraints and variables, so a straightforward computer application of the simplex method may require an exorbitant computational effort. Fortunately, another characteristic is that most of the a_{ij} coefficients in the constraints are zeroes, and the relatively few nonzero coefficients appear in a distinctive pattern. As a result, it has been possible to develop special *streamlined* versions of the simplex method that achieve dramatic computational savings by exploiting this *special structure* of the problem. Therefore, it is important to become sufficiently familiar with these special types of problems so that you can recognize them when they arise and apply the proper computational procedure.

To describe special structures, we shall again use the table (matrix) of constraint coefficients, first shown in Table 8.1 and repeated here in Table 23.1, where a_{ij} is the coefficient of the jth variable in the ith functional constraint. Later, portions of the table containing only coefficients equal to zero will be indicated by leaving them blank, whereas blocks containing nonzero coefficients will be shaded darker.

The first section presents the *transshipment problem,* which is both an extension of the transportation problem and a special case of the minimum cost flow problem.

Sections 23.2 to 23.5 discuss some special types of linear programming problems that can be characterized by where the *blocks of nonzero coefficients* appear in the table of constraint coefficients. One type frequently arises in multidivisional organizations. A second arises in multitime period problems. A third combines the first two types. Section 23.3 describes the *decomposition principle* for streamlining the simplex method to efficiently solve either the first type or the dual of the second type.

TABLE 23.1 Table of constraint coefficients for linear programming

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

One of the practical problems involved in the application of linear programming is the uncertainty about what the values of the model parameters will turn out to be when the adopted solution actually is implemented. Occasionally, the degree of uncertainty is so great that some or all of the model parameters need to be treated explicitly as *random variables.* Sections 23.6 and 23.7 present two special formulations, *stochastic programming* and *chance-constrained programming,* for this problem of *linear programming under uncertainty.*

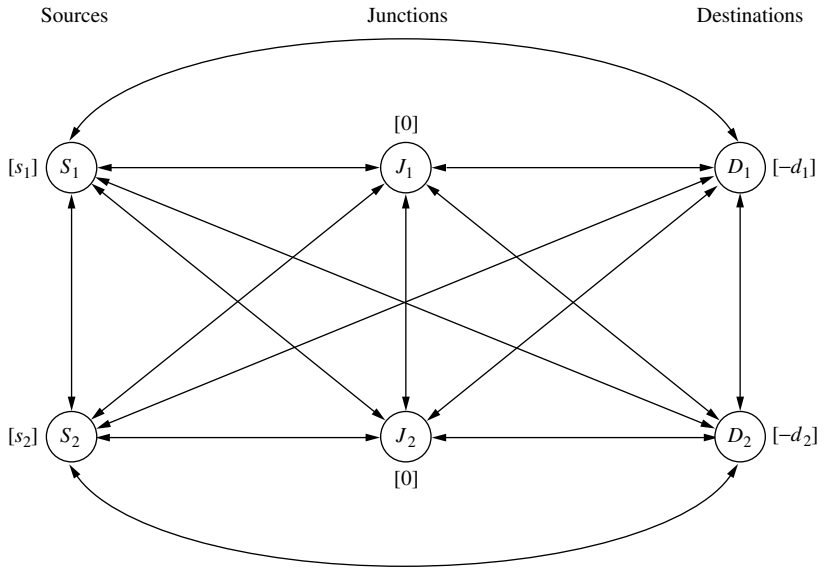
23.1 THE TRANSSHIPMENT PROBLEM

One requirement of the transportation problem presented in Sec. 8.1 is advance knowledge of the method of distribution of units from each source i to each destination j, so that the corresponding cost per unit (c_{ij}) can be determined. Sometimes, however, the best method of distribution is not clear because of the possibility of **transshipments,** whereby shipments would go through intermediate transfer points (which might be other sources or destinations). For example, rather than shipping a special cargo directly from port 1 to port 3, it may be cheaper to include it with regular cargoes from port 1 to port 2 and then from port 2 to port 3.

Such possibilities for transshipments could be investigated in advance to determine the cheapest route from each source to each destination. However, this might be a very complicated and time-consuming task if there are many possible intermediate transfer points. Therefore, it may be much more convenient to let a computer algorithm solve *simultaneously* for the amount to ship from each source to each destination *and* the route to follow for each shipment so as to minimize the total shipping cost.

This extension of the transportation problem to include the routing decisions is referred to as the **transshipment problem.** This problem is the special case of the minimum cost flow problem presented in Sec. 9.6 where there are no restrictions on the amount that can be shipped through each shipping lane (unlimited arc capacities). The network representation of such a problem is displayed in Fig. 23.1, where each two-sided arrow indicates that a shipment can be sent in either direction between the corresponding pair of locations. To avoid undue clutter, this network shows only the first two sources, destinations, and *junctions* (intermediate transfer points that are neither sources nor destinations), and the unit shipping cost associated with each arrow has been deleted. (As in Figs. 8.2 and 8.3, the quantity in square brackets next to each location is the net number of units to be shipped out of that location). Even when showing only these few locations, note that there now are many possible routes for a shipment from any particular source to any particular destination, including through other sources or destinations en route. With a large network, finding the cheapest such route is not an easy task.

Fortunately, there is a simple way to reformulate the transshipment problem to fit it back into the format of the transportation problem. Thus, the *transportation simplex method* presented in Sec. 8.2 can be used to solve the transshipment problem. (As a special case of the minimum cost flow problem, the transshipment problem also can be solved by the *network simplex method* described in Sec. 9.7.)



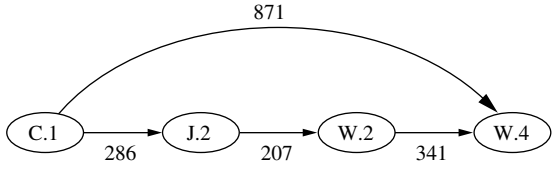
■ **FIGURE 23.1**
The network representation of the transshipment problem.

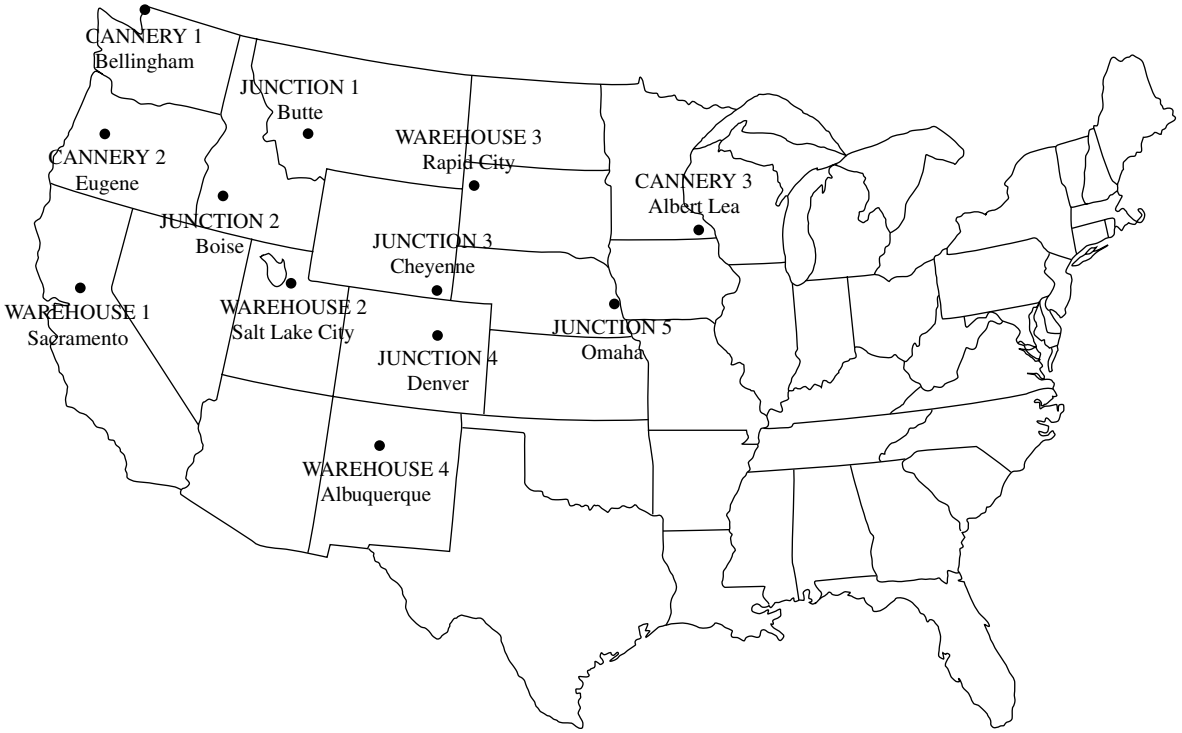
To clarify the structure of the transshipment problem and the nature of this reformulation, we shall now extend the prototype example for the transportation problem to include transshipments.

Prototype Example

After further investigation, the P & T COMPANY (see Sec. 8.1) has found that it can cut costs by discontinuing its own trucking operation and using common carriers instead to truck its canned peas. Since no single trucking company serves the entire area containing all the canneries and warehouses, many of the shipments will need to be transferred to another truck at least once along the way. These transfers can be made at intermediate canneries or warehouses, or at five other locations (Butte, Montana; Boise, Idaho; Cheyenne, Wyoming; Denver, Colorado; and Omaha, Nebraska) referred to as junctions, as shown in Fig. 23.2. The shipping cost per truckload between each of these points is given in Table 23.2, where a dash indicates that a direct shipment is not possible.

For example, a truckload of peas can still be sent from cannery 1 to warehouse 4 by direct shipment at a cost of $871. However, another possibility, shown below, is to ship the truckload from cannery 1 to junction 2, transfer it to a truck going to warehouse 2, and then transfer it again to go to warehouse 4, at a cost of only ($286 + $207 + $341) = $834.





■ **FIGURE 23.2**
Location of canneries, warehouses, and junctions for the P & T Co.

■ **TABLE 23.2** Independent trucking data for P & T Co.

		Shipping Cost per Truckload											
	To	Cannery			Junction					Warehouse			
From		1	2	3	1	2	3	4	5	1	2	3	4	Output
	1		$146	—	$324	$286	—	—	—	$452	$505	—	$871	75
Cannery	2	$146		—	$373	$212	$570	$609	—	$335	$407	$688	$784	125
	3	—	—		$658	—	$405	$419	$158	—	$685	$359	$673	100
	1	$322	$371	$656		$262	$398	$430	—	$503	$234	$329	—	
	2	$284	$210	—	$262		$406	$421	$644	$305	$207	$464	$558	
Junction	3	—	$569	$403	$398	$406		$ 81	$272	$597	$253	$171	$282	
	4	—	$608	$418	$431	$422	$ 81		$287	$613	$280	$236	$229	
	5	—	—	$158	—	$647	$274	$288		$831	$501	$293	$482	
	1	$453	$336	—	$505	$307	$599	$615	$831		$359	$706	$587	
Warehouse	2	$505	$407	$683	$235	$208	$254	$281	$500	$357		$362	$341	
	3	—	$687	$357	$329	$464	$171	$236	$290	$705	$362		$457	
	4	$868	$781	$670	—	$558	$282	$229	$480	$587	$340	$457		
Allocation										80	65	70	85	

This possibility is only one of many indirect ways of shipping a truckload from cannery 1 to warehouse 4 that needs to be considered, if indeed this cannery should send anything to this warehouse. The overall problem is to determine how the output from all the canneries should be shipped to meet the warehouse allocations and minimize the total shipping cost.

Now let us see how this transshipment problem can be reformulated as a transportation problem. The basic idea is to interpret the individual truck trips (as opposed to complete journeys for truckloads) as being the shipment from a source to a destination, and so label *all* 12 locations (canneries, junctions, and warehouses) as being both potential *destinations* and potential *sources* for these shipments. To illustrate this interpretation, consider the above example where a truckload of peas is shipped from cannery 1 to warehouse 4 by being *transshipped* through junction 2 and then warehouse 2. The first truck trip for this shipment has cannery 1 as its source and junction 2 as its destination, but then junction 2 becomes the source for the second truck trip with warehouse 2 as its destination. Finally, warehouse 2 becomes the source for the third trip with this same shipment, where warehouse 4 then is the destination. In a similar fashion, any of the 12 locations can become a source, a destination, or both, for truck trips.

Thus, for the reformulation as a transportation problem, we have 12 sources and 12 destinations. The c_{ij} unit costs for the resulting *parameter table* shown in Table 23.3 are just the shipping costs per truckload already given in Table 23.2. The impossible shipments indicated by dashes in Table 23.2 are assigned a huge unit cost of M. Because each location is both a source and a destination, the diagonal elements in the parameter table represent the unit cost of a shipment from a given location *to itself*. The costs of these fictional shipments going nowhere are zero.

To complete the reformulation of this transshipment problem as a transportation problem, we now need to explain how to obtain the demand and supply quantities in Table 23.3. The number of truckloads transshipped through a location should be included in both the demand for that location as a destination and the supply for that location as a source. Since we do not know this number in advance, we instead add a safe upper bound on this number to both the original demand and supply for that location (shown as allocation and output

TABLE 23.3 Parameter Table for the P & T Co. transshipment problem formulated as a transportation problem

			Destination											
			(Canneries)			(Junctions)					(Warehouses)			
			1	2	3	4	5	6	7	8	9	10	11	12	Supply
		1	0	146	*M*	324	286	*M*	*M*	*M*	452	505	*M*	871	375
	(Canneries)	2	146	0	*M*	373	212	570	609	*M*	335	407	688	784	425
		3	*M*	*M*	0	658	*M*	405	419	158	*M*	685	359	673	400
		4	322	371	656	0	262	398	430	*M*	503	234	329	*M*	300
		5	284	210	*M*	262	0	406	421	644	305	207	464	558	300
Source	(Junctions)	6	*M*	569	403	398	406	0	81	272	597	253	171	282	300
		7	*M*	608	418	431	422	81	0	287	613	280	236	229	300
		8	*M*	*M*	158	*M*	647	274	288	0	831	501	293	482	300
		9	453	336	*M*	505	307	599	615	831	0	359	706	587	300
		10	505	407	683	235	208	254	281	500	357	0	362	341	300
	(Warehouses)	11	*M*	687	357	329	464	171	236	290	705	362	0	457	300
		12	868	781	670	*M*	558	282	229	480	587	340	457	0	300
Demand			300	300	300	300	300	300	300	300	380	365	370	385	

in Table 23.2) and then introduce the same slack variable into its demand and supply constraints. This single slack variable thereby serves the role of both a dummy source and a dummy destination.) Since it would never pay to return a truckload to be transshipped through the same location more than once, a safe upper bound on this number for any location is the *total number of truckloads* (300), so we shall use 300 as the upper bound. The slack variable for both constraints for location i would be x_{ii}, the (fictional) number of truckloads shipped from this location to itself. Thus, $(300 - x_{ii})$ is the real number of truckloads transshipped through location i.

Adding 300 to each of the allocation and demand quantities in Table 23.2 (where blanks are zeros) now gives us the complete parameter table shown in Table 23.3 for the transportation problem formulation of our transshipment problem. Therefore, using the transportation simplex method to obtain an optimal solution for this transportation problem provides an optimal shipping plan (ignoring the x_{ii}) for the P & T Company.

General Features

Our prototype example illustrates all the general features of the transshipment problem and its relationship to the transportation problem. Thus, the transshipment problem can be described in general terms as being concerned with how to allocate and route units (truckloads of canned peas in the example) from *supply centers* (canneries) to *receiving centers* (warehouses) via intermediate *transshipment points* (junctions, other supply centers, and other receiving centers). (The network representation in Fig. 23.1 ignores the geographical layout of these locations by lining up all the supply centers in the first column, all the junctions in the second column, and all the receiving centers in the third column.) In addition to transshipping units, each supply center generates a given net surplus of units to be distributed, and each receiving center absorbs a given net deficit, whereas each junction neither generates nor absorbs any units. (The net number of units generated at each location is shown in square brackets next to that location in Fig. 23.1.) The problem has feasible solutions only if the total net surplus generated at the supply centers *equals* the total net deficit to be absorbed at the receiving centers.

A direct shipment may be impossible ($c_{ij} = M$) for certain pairs of locations. In addition, certain supply centers and receiving centers may not be able to serve as transshipment points at all. In the reformulation of the transshipment problem as a transportation problem, the easiest way to deal with any such center is to delete its column (for a supply center) or its row (for a receiving center) in the parameter table, and then add nothing to its original supply or demand quantity.

A positive cost c_{ij} is incurred for each unit sent *directly* from location i (a supply center, junction, or receiving center) to another location j. The objective is to determine the plan for allocating and routing the units that minimizes the total cost.

The resulting mathematical model for the transshipment problem (see Prob. 23.1-4) has a special structure slightly different from that for the transportation problem. As in the latter case, it has been found that some applications that have nothing to do with transportation can be fitted to this special structure. However, regardless of the physical context of the application, this model always can be reformulated as an equivalent transportation problem in the manner illustrated by the prototype example.

This reformulation is not necessary to solve a transshipment problem. Another alternative is to apply the network simplex method (see Sec. 9.7) to the problem directly without any reformulation. Even though the transportation simplex method (see Sec. 8.2) is a little more efficient than the network simplex method for solving transportation problems, the great efficiency of the network simplex method in general makes this a reasonable alternative.

23.2 MULTIDIVISIONAL PROBLEMS

Another important class of linear programming problems having an exploitable special structure consists of **multidivisional problems.** Their special feature is that they involve coordinating the decisions of the separate divisions of a large organization. Because the divisions operate with considerable autonomy, the problem is *almost* decomposable into separate problems, where each division is concerned only with optimizing its own operation. However, some overall coordination is required in order to best divide certain organizational resources among the divisions.

As a result of this special feature, the table of constraint coefficients for multidivisional problems has the **block angular structure** shown in Table 23.4. (Recall that shaded blocks represent the only portions of the table that have *any* nonzero a_{ij} coefficients.) Thus, each smaller block contains the coefficients of the constraints for one **subproblem,** namely, the problem of optimizing the operation of a division considered by itself. The long block at the top gives the coefficients of the **linking constraints** for the **master problem,** namely, the problem of coordinating the activities of the divisions by dividing organizational resources among them so as to obtain an overall optimal solution for the entire organization.

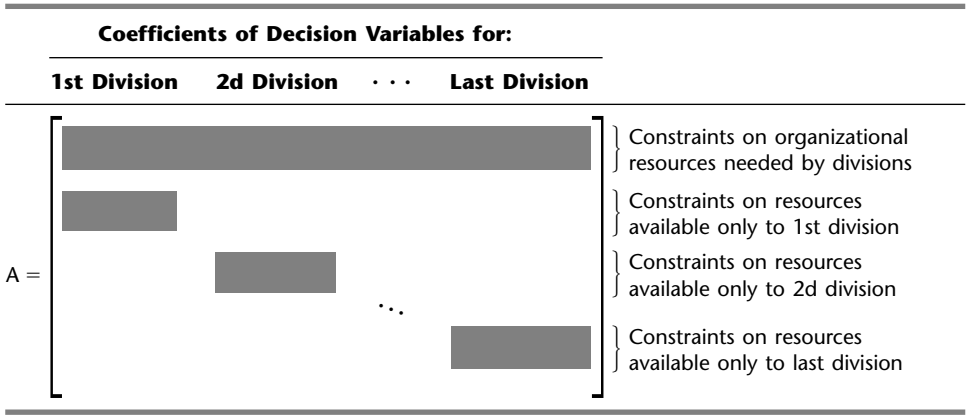
Because of their nature, multidivisional problems frequently are very large, containing many thousands of constraints and variables. Therefore, it may be necessary to exploit the special structure in order to be able to solve such a problem with a reasonable expenditure of computer time, or even to solve it at all! The **decomposition principle** (described in Sec. 23.3) provides an effective way of exploiting the special structure.

Conceptually, this streamlined version of the simplex method can be thought of as having each division solve its subproblem and sending this solution as its proposal to "headquarters" (the master problem), where negotiators then coordinate the proposals from all the divisions to find an optimal solution for the overall organization. If the subproblems are of manageable size and the master problem is not too large (not more than 50 to 100 constraints), this approach is successful in solving some *extremely* large multidivisional problems. It is particularly worthwhile when the total number of constraints is quite large (at least several thousand) and there are more than a few subproblems.

Prototype Example

The GOOD FOODS CORPORATION is a very large producer and distributor of food products. It has three main divisions: the Processed Foods Division, the Canned Foods Division, and the Frozen Foods Division. Because costs and market prices change frequently

TABLE 23.4 Constraint coefficients for multidivisional problems



in the food industry, Good Foods periodically uses a corporate linear programming model to revise the production rates for its various products in order to use its available production capacities in the most profitable way. This model is similar to that for the Wyndor Glass Co. problem (see Sec. 3.1), but on a much larger scale, having thousands of constraints and variables. (Since our space is limited, we shall describe a simplified version of this model that combines the products or resources by types.)

The corporation grows its own high-quality corn and potatoes, and these basic food materials are the only ones currently in short supply that are used by all the divisions. Except for these organizational resources, each division uses only its own resources and thus could determine its optimal production rates autonomously. The data for each division and the corresponding subproblem involving just its products and resources are given in Table 23.5 (where Z represents profit in millions of dollars per month), along with the data for the organizational resources.

The resulting linear programming problem for the corporation is

$$\text{Maximize} \qquad Z = 8x_1 + 5x_2 + 6x_3 + 9x_4 + 7x_5 + 9x_6 + 6x_7 + 5x_8,$$

subject to

$$\begin{aligned}
5x_1 + 3x_2 \qquad\quad + 2x_4 \qquad\quad + 3x_6 + 4x_7 + 6x_8 &\le 30 \\
2x_1 \qquad + 4x_3 + 3x_4 + 7x_5 \qquad\quad + \ x_7 \qquad\quad &\le 20 \\
2x_1 + 4x_2 + 3x_3 \qquad\quad\quad\quad\quad\quad\quad\quad &\le 10 \\
7x_1 + 3x_2 + 6x_3 \qquad\quad\quad\quad\quad\quad\quad\quad &\le 15 \\
5x_1 \qquad + 3x_3 \qquad\quad\quad\quad\quad\quad\quad\quad &\le 12 \\
3x_4 + \ x_5 + 2x_6 \qquad\quad\quad\quad &\le \ 7 \\
2x_4 + 4x_5 + 3x_6 \qquad\quad\quad\quad &\le \ 9 \\
8x_7 + 5x_8 &\le 25 \\
7x_7 + 9x_8 &\le 30 \\
6x_7 + 4x_8 &\le 20
\end{aligned}$$

and

$$x_j \ge 0, \qquad \text{for } j = 1, 2, \ldots, 8.$$

Note how the corresponding table of constraint coefficients shown in Table 23.6 fits the special structure for multidivisional problems given in Table 23.4. Therefore, the Good Foods Corp. can indeed solve this problem (or a more detailed version of it) by the streamlined version of the simplex method provided by the decomposition principle.

Important Special Cases

Some even simpler forms of the special structure exhibited in Table 23.4 arise quite frequently. Two particularly common forms are shown in Table 23.7.

The first form occurs when some or all of the variables can be divided into groups such that the sum of the variables in each group must not exceed a specified upper bound for that group (or perhaps must equal a specified constant). Constraints of this form,

$$\begin{aligned}
&x_{j1} + x_{j2} + \cdots + x_{jk} \le b_i \\
(\text{or} \quad &x_{j1} + x_{j2} + \cdots + x_{jk} = b_i),
\end{aligned}$$

usually are called either *generalized upper-bound constraints* (**GUB constraints** for short) or *group constraints.* Although Table 23.7 shows each GUB constraint as involving consecutive variables, this is not necessary. For example,

$$x_1 + x_5 = x_9 \le 1$$

is a GUB constraint, as is

$$x_8 + x_3 + x_6 = 20.$$

TABLE 23.5 Data for the Good Foods Corp. multidivisional problem

Divisional Data

Processed Foods Division

Resource \ Product	Resource Usage/Unit 1	2	3	Amount Available
1	2	4	3	10
2	7	3	6	15
3	5	0	3	12
ΔZ/unit	8	5	6	
Level	x_1	x_2	x_3	

Canned Foods Division

Resource \ Product	Resource Usage/Unit 4	5	6	Amount Available
4	3	1	2	7
5	2	4	3	9
ΔZ/unit	9	7	9	
Level	x_4	x_5	x_6	

Frozen Foods Division

Resource \ Product	Resource Usage/Unit 7	8	Amount Available
6	8	5	25
7	7	9	30
8	6	4	20
ΔZ/unit	6	5	
Level	x_7	x_8	

Subproblem

Maximize $Z_1 = 8x_1 + 5x_2 + 6x_3$,

subject to

$$2x_1 + 4x_2 + 3x_3 \leq 10$$
$$7x_1 + 3x_2 + 6x_3 \leq 15$$
$$5x_1 + 3x_3 \leq 12$$

and $x_1 \geq 0, \quad x_2 \geq 0, \quad x_3 \geq 0.$

Maximize $Z_2 = 9x_4 + 7x_5 + 9x_6$,

subject to

$$3x_4 + x_5 + 2x_6 \leq 7$$
$$2x_4 + 4x_5 + 3x_6 \leq 9$$

and $x_4 \geq 0, \quad x_5 \geq 0, \quad x_6 \geq 0.$

Maximize $Z_3 = 6x_7 + 5x_8$,

subject to

$$8x_7 + 5x_8 \leq 25$$
$$7x_7 + 9x_8 \leq 30$$
$$6x_7 + 4x_8 \leq 20$$

and $x_7 \geq 0, \quad x_8 \geq 0.$

Data for Organizational Resources

Resource \ Product	Resource Usage/Unit 1	2	3	4	5	6	7	8	Amount Available
Corn	5	3	0	2	0	3	4	6	30
Potatoes	2	0	4	3	7	0	1	0	20

The second form shown in Table 23.7 occurs when some or all of the individual variables must not exceed a specified upper bound for that variable. These constraints,

$$x_j \leq b_i,$$

normally are referred to as **upper-bound constraints.** For example, both

$$x_1 \leq 1 \qquad \text{and} \qquad x_2 \leq 5$$

are upper-bound constraints. A special technique for dealing efficiently with such constraints has been described in Sec. 7.3.

TABLE 23.6 Constraint coefficients for the Good Foods Corp. multidivisional problem

TABLE 23.7 Constraint coefficients for important special cases of the structure for multidivisional problems given in Table 23.4

Either GUB or upper-bound constraints may occur because of the multidivisional nature of the problem. However, we should emphasize that they often arise in many other contexts as well. In fact, you already have seen several examples containing such constraints as summarized below.

Note in Table 8.6 that all supply constraints in the transportation problem actually are GUB constraints. (Table 8.6 fits the form in Table 23.7 by placing the supply constraints below the demand constraints.) In addition, the demand constraints also are GUB constraints, but ones not involving *consecutive* variables.

In the Southern Confederation of Kibbutzim regional planning problem (see Sec. 3.4), the constraints involving usable land for each kibbutz and total acreage for each crop all are GUB constraints.

The technological limit constraints in the Nori & Leets Co. air pollution problem (see Sec. 3.4) are upper-bound constraints, as are two of the three functional constraints in the Wyndor Glass Co. product mix problem (see Sec. 3.1).

Because of the prevalence of GUB and upper-bound constraints, it is very helpful to have special techniques for streamlining the way in which the simplex method deals with them.

(The technique for GUB constraints[1] is quite similar to the one for upper-bound constraints described in Sec. 7.3.) If there are many such constraints, these techniques can drastically reduce the computation time for a problem.

23.3 THE DECOMPOSITION PRINCIPLE FOR MULTIDIVISIONAL PROBLEMS

In Sec. 23.2, we discussed the special class of linear programming problems called *multidivisional problems* and their special block angular structure (see Table 23.4). We also mentioned that the streamlined version of the simplex method called the *decomposition principle* provides an effective way of exploiting this special structure to solve very large problems. (This approach also is applicable to the dual of the class of multitime period problems presented in Sec. 23.4.) We shall describe and illustrate this procedure after reformulating (decomposing) the problem in a way that enables the algorithm to exploit its special structure.

A Useful Reformulation (Decomposition) of the Problem

The basic approach is to reformulate the problem in a way that greatly reduces the number of functional constraints and then to apply the *revised simplex method* (see Sec. 5.2). Therefore, we need to begin by giving the *matrix form* of multidivisional problems:

Maximize $\quad Z = \mathbf{cx},$

subject to

$\mathbf{Ax} \leq \mathbf{b}$† $\quad$ and $\quad \mathbf{x} \leq \mathbf{0},$

where the $\mathbf{A}$ matrix has the block angular structure

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_1 & \mathbf{A}_2 & \cdots & \mathbf{A}_N \\ \mathbf{A}_{N+1} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{N+2} & \cdots & \mathbf{0} \\ \vdots & \vdots & & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{A}_{2N} \end{bmatrix}$$

where the $\mathbf{A}_i$ ($i = 1, 2, \ldots, 2N$) are matrices, and the $\mathbf{0}$ are null matrices. Expanding, this can be rewritten as

Maximize $\quad Z = \sum_{j=1}^{N} \mathbf{c}_j\mathbf{x}_j,$

subject to

$$[\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_N, \mathbf{I}]\begin{bmatrix} \mathbf{x} \\ \mathbf{x}_s \end{bmatrix} = \mathbf{b}_0, \qquad \begin{bmatrix} \mathbf{x} \\ \mathbf{x}_s \end{bmatrix} \geq \mathbf{0},$$

$$\mathbf{A}_{N+j}\mathbf{x}_j \leq \mathbf{b}_j \quad \text{and} \quad \mathbf{x}_j \geq \mathbf{0}, \quad \text{for } j = 1, 2, \ldots, N,$$

[1]G. B. Dantzig, and R. M. Van Slyke, "Generalized Upper Bounded Techniques for Linear Programming," *Journal of Computer and Systems Sciences,* **1:** 213–226, 1967.

†The following discussion would not be changed substantially if $\mathbf{Ax} = \mathbf{b}$.

where $\mathbf{c}_j$, $\mathbf{x}_j$, $\mathbf{b}_0$, and $\mathbf{b}_j$ are vectors such that $\mathbf{c} = [\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_N]$,

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_N \end{bmatrix}, \qquad \mathbf{b} = \begin{bmatrix} \mathbf{b}_0 \\ \mathbf{b}_1 \\ \vdots \\ \mathbf{b}_N \end{bmatrix},$$

and where $\mathbf{x}_s$ is the vector of slack variables for the first set of constraints.

This structure suggests that it may be possible to solve the overall problem by doing little more than solving the N subproblems of the form

$$\text{Maximize} \qquad Z_j = \mathbf{c}_j\mathbf{x}_j,$$

subject to

$$\mathbf{A}_{N+j}\mathbf{x}_j \leq \mathbf{b}_j \qquad \text{and} \qquad \mathbf{x}_j \geq 0,$$

thereby greatly reducing computational effort. After some reformulation, this approach can indeed be used.

Assume that the set of feasible solutions for each subproblem is a bounded set (i.e., none of the variables can approach infinity). Although a more complicated version of the approach can still be used otherwise, this assumption will simplify the discussion.

The set of points $\mathbf{x}_j$ such that $\mathbf{x}_j \geq \mathbf{0}$ and $\mathbf{A}_{N+j}\mathbf{x}_j \leq \mathbf{b}_j$ constitutes a *convex set* with a finite number of *extreme points* (the CPF solutions for the subproblem having these constraints.)[1] Therefore, under the assumption that the set is bounded, any point in the set can be represented as a convex combination of the extreme points. To express this mathematically, let n_j be the number of extreme points, and denote these points by $\mathbf{x}_{jk}^*$ for $k = 1, 2, \ldots, n_j$. Then any solution $\mathbf{x}_j$ to subproblem j that satisfies the constraints $\mathbf{A}_{N+j}\mathbf{x}_j \leq \mathbf{b}_j$ and $\mathbf{x}_j \geq 0$ also satisfies the equation

$$\mathbf{x}_j = \sum_{k=1}^{n_j} \rho_{jk}\mathbf{x}_{jk}^*$$

for some combination of ρ_{jk} such that

$$\sum_{k=1}^{n_j} \rho_{jk} = 1$$

and $\rho_{jk} \geq 0$ ($k = 1, 2, \ldots, n_j$). Furthermore, this is not true for any $\mathbf{x}_j$ that is not a feasible solution for subproblem j. (You may have shown these facts for Prob. 4.5-5.)

Therefore, this equation for $\mathbf{x}_j$ and the constraints on the ρ_{jk} provide a method for representing the feasible solutions to subproblem j without using any of the original constraints. Hence, the overall problem can now be reformulated with far fewer constraints as

$$\text{Maximize} \qquad Z = \sum_{j=1}^{N} \sum_{k=1}^{n_j} (\mathbf{c}_j\mathbf{x}_{jk}^*)\rho_{jk},$$

subject to

$$\sum_{j=1}^{N} \sum_{k=1}^{n_j} (\mathbf{A}_j\mathbf{x}_{jk}^*)\rho_{jk} + \mathbf{x}_s = \mathbf{b}_0, \ \mathbf{x}_s \geq \mathbf{0}, \qquad \sum_{k=1}^{n_j} \rho_{jk} = 1, \qquad \text{for } j = 1, 2, \ldots, N,$$

[1]See Appendix 2 for a definition and discussion of convex sets and extreme points.

and

$$\rho_{jk} \geq 0, \qquad \text{for } j = 1, 2, \ldots, N \qquad \text{and} \qquad k = 1, 2, \ldots, n_j.$$

This formulation is completely equivalent to the one given earlier. However, since it has far fewer constraints, it should be solvable with much less computational effort. The fact that the number of variables (which are now the ρ_{jk} and the elements of $\mathbf{x}_s$) is much larger does not matter much computationally if the revised simplex method is used. The one apparent flaw is that it would be tedious to identify all the $\mathbf{x}_{jk}^*$. Fortunately, it is not necessary to do this when using the revised simplex method. The procedure is outlined below.

The Algorithm Based on This Decomposition

Let $\mathbf{A}'$ be the matrix of constraint coefficients for this reformulation of the problem, and let $\mathbf{c}'$ be the vector of objective function coefficients. (The individual elements of $\mathbf{A}'$ and $\mathbf{c}'$ are determined only when they are needed.) As usual, let $\mathbf{B}$ be the current basis matrix, and let $\mathbf{c}_B$ be the corresponding vector of basic variable coefficients in the objective function.

For a portion of the work required for the optimality test and step 1 of an iteration, the revised simplex method needs to find the minimum element of $(\mathbf{c}_B\mathbf{B}^{-1}\mathbf{A}' - \mathbf{c}')$, the vector of coefficients of the original variables (the ρ_{jk} in this case) in the current Eq. (0). Let $(z_{jk} - c_{jk})$ denote the element in this vector corresponding to ρ_{jk}. Let m_0 denote the number of elements of $\mathbf{b}_0$. Let $(\mathbf{B}^{-1})_{1;m_0}$ be the matrix consisting of the first m_0 columns of $\mathbf{B}^{-1}$, and let $(\mathbf{B}^{-1})_i$ be the vector consisting of the ith column of $\mathbf{B}^{-1}$. Then $(z_{jk} - c_{jk})$ reduces to

$$\begin{aligned} z_{jk} - c_{jk} &= \mathbf{c}_B(\mathbf{B}^{-1})_{1;m_0}\mathbf{A}_j\mathbf{x}_{jk}^* + \mathbf{c}_B(\mathbf{B}^{-1})_{m_0+j} - \mathbf{c}_j\mathbf{x}_{jk}^* \\ &= (\mathbf{c}_B(\mathbf{B}^{-1})_{1;m_0}\mathbf{A}_j - \mathbf{c}_j)\mathbf{x}_{jk}^* + \mathbf{c}_B(\mathbf{B}^{-1})_{m_0+j}. \end{aligned}$$

Since $\mathbf{c}_B(\mathbf{B}^{-1})_{m_0+j}$ is independent of k, the *minimum* value of $(z_{jk} - c_{jk})$ over $k = 1, 2, \ldots, n_j$ can be found as follows. The $\mathbf{x}_{jk}^*$ are just the CPF solutions for the set of constraints, $\mathbf{x}_j \geq \mathbf{0}$ and $\mathbf{A}_{N+j}\mathbf{x}_j \leq \mathbf{b}_j$, and the simplex method identifies the CPF solution that minimizes (or maximizes) a given objective function. Therefore, solve the linear programming problem

$$\text{Minimize} \qquad W_j = (\mathbf{c}_B(\mathbf{B}^{-1})_{1;m_0}\mathbf{A}_j - \mathbf{c}_j)\mathbf{x}_j + \mathbf{c}_B(\mathbf{B}^{-1})_{m_0+j},$$

subject to

$$\mathbf{A}_{N+j}\mathbf{x}_j \leq \mathbf{b}_j \qquad \text{and} \qquad \mathbf{x}_j \geq \mathbf{0}.$$

The optimal value of W_j (denoted by W_j^*) is the desired minimum value of $(z_{jk} - c_{jk})$ over k. Furthermore, the optimal solution for $\mathbf{x}_j$ is the corresponding $\mathbf{x}_{jk}^*$.

Therefore, the first step at each iteration requires solving N linear programming problems of the above type to find W_j^* for $j = 1, 2, \ldots, N$. In addition, the current Eq. (0) coefficients of the elements of $\mathbf{x}_s$ that are nonbasic variables would be found in the usual way as the elements of $\mathbf{c}_B(\mathbf{B}^{-1})_{1;m_0}$. If all these coefficients [the W_j^* and the elements of $\mathbf{c}_B(\mathbf{B}^{-1})_{1;m_0}$] are nonnegative, the current solution is optimal by the optimality test. Otherwise, the minimum of these coefficients is found, and the corresponding variable is selected as the new entering basic variable. If that variable is ρ_{jk}, then the solution to the linear programming problem involving W_j has identified $\mathbf{x}_{jk}^*$, so that the original constraint coefficients of ρ_{jk} are now identified. Hence, the revised simplex method can complete the iteration in the usual way.

Assuming that $\mathbf{x} = \mathbf{0}$ is feasible for the original problem, the initialization step would use the corresponding solution in the reformulated problem as the initial BF solution. This

involves selecting the initial set of basic variables (the elements of $\mathbf{x}_B$) to be the elements of $\mathbf{x}_s$ and the one variable ρ_{jk} for each subproblem j ($j = 1, 2, \ldots, N$) such that $\mathbf{x}_{jk}^* = \mathbf{0}$. Following the initialization step, the above procedure is repeated for a succession of iterations until an optimal solution is reached. The optimal values of the ρ_{jk} are then substituted into the equations for the $\mathbf{x}_j$ for the optimal solution to conform to the original form of the problem.

Example. To illustrate this procedure, consider the problem

$$\text{Maximize} \qquad Z = 4x_1 + 6x_2 + 8x_3 + 5x_4,$$

subject to

$$\begin{aligned} x_1 + 3x_2 + 2x_3 + 4x_4 &\le 20 \\ 2x_1 + 3x_2 + 6x_3 + 4x_4 &\le 25 \\ x_1 + x_2 &\le 5 \\ x_1 + 2x_2 &\le 8 \\ 4x_3 + 3x_4 &\le 12 \end{aligned}$$

and

$$x_j \ge 0, \qquad \text{for } j = 1, 2, 3, 4.$$

Thus, the $\mathbf{A}$ matrix is

$$\mathbf{A} = \left[\begin{array}{cc:cc} 1 & 3 & 2 & 4 \\ 2 & 3 & 6 & 4 \\ \hdashline 1 & 1 & 0 & 0 \\ 1 & 2 & 0 & 0 \\ \hdashline 0 & 0 & 4 & 3 \end{array}\right],$$

so that $N = 2$ and

$$\mathbf{A}_1 = \begin{bmatrix} 1 & 3 \\ 2 & 3 \end{bmatrix}, \qquad \mathbf{A}_2 = \begin{bmatrix} 2 & 4 \\ 6 & 4 \end{bmatrix}, \qquad \mathbf{A}_3 = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}, \qquad \mathbf{A}_4 = [4,\ 3].$$

In addition,

$$\mathbf{c}_1 = [4, 6], \qquad \mathbf{c}_2 = [8, 5],$$

$$\mathbf{x}_1 = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \qquad \mathbf{x}_2 = \begin{bmatrix} x_3 \\ x_4 \end{bmatrix}, \qquad \mathbf{b}_0 = \begin{bmatrix} 20 \\ 25 \end{bmatrix}, \qquad \mathbf{b}_1 = \begin{bmatrix} 5 \\ 8 \end{bmatrix}, \qquad \mathbf{b}_2 = [12].$$

To prepare for demonstrating how this problem would be solved, we shall first examine its two subproblems individually and then construct the reformulation of the overall problem. Thus, *subproblem 1* is

$$\text{Maximize} \qquad Z_1 = [4, 6]\begin{bmatrix} x_1 \\ x_2 \end{bmatrix},$$

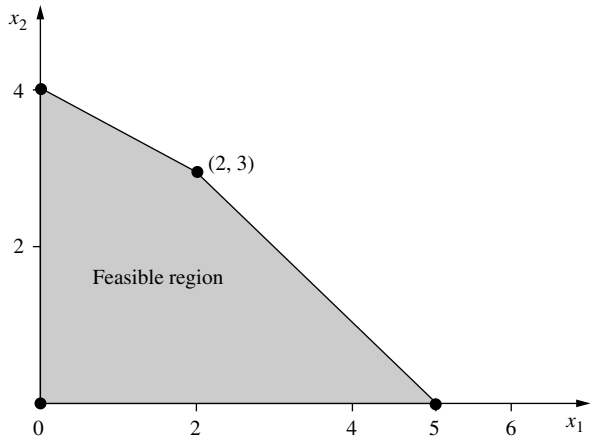
subject to

$$\begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \le \begin{bmatrix} 5 \\ 8 \end{bmatrix} \qquad \text{and} \qquad \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \ge \begin{bmatrix} 0 \\ 0 \end{bmatrix},$$

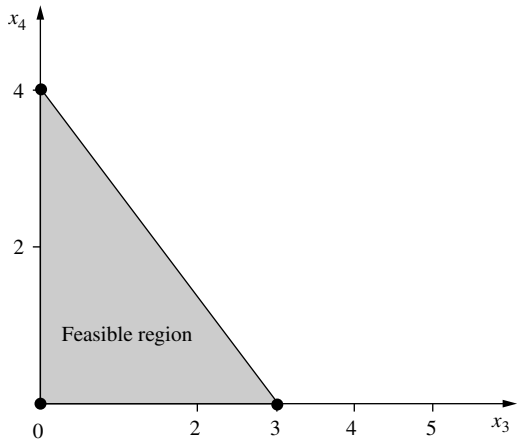
so that its set of feasible solutions is as shown in Fig. 23.3.

It can be seen that this subproblem has four extreme points ($n_1 = 4$), namely, the four CPF solutions shown by dots in Fig. 23.3. One of these is the origin, considered the "first" of these extreme points, so

$$\mathbf{x}_{11}^* = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \qquad \mathbf{x}_{12}^* = \begin{bmatrix} 5 \\ 0 \end{bmatrix}, \qquad \mathbf{x}_{13}^* = \begin{bmatrix} 2 \\ 3 \end{bmatrix}, \qquad \mathbf{x}_{14}^* = \begin{bmatrix} 0 \\ 4 \end{bmatrix},$$



■ **FIGURE 23.3**
Subproblem 1 for the example illustrating the decomposition principle.



■ **FIGURE 23.4**
Subproblem 2 for the example illustrating the decomposition principle.

where ρ_{11}, ρ_{12}, ρ_{13}, ρ_{14} are the respective weights on these points.

Similarly, *subproblem* 2 is

$$\text{Maximize} \qquad Z_2 = [8, 5] \begin{bmatrix} x_3 \\ x_4 \end{bmatrix},$$

subject to

$$[4, 3] \begin{bmatrix} x_3 \\ x_4 \end{bmatrix} \leq [12] \qquad \text{and} \qquad \begin{bmatrix} x_3 \\ x_4 \end{bmatrix} \geq \begin{bmatrix} 0 \\ 0 \end{bmatrix},$$

and its set of feasible solutions is shown in Fig. 23.4. Thus, its three extreme points are

$$\mathbf{x}_{21}^* = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \qquad \mathbf{x}_{22}^* = \begin{bmatrix} 3 \\ 0 \end{bmatrix}, \qquad \mathbf{x}_{23}^* = \begin{bmatrix} 0 \\ 4 \end{bmatrix},$$

where ρ_{21}, ρ_{22}, ρ_{23} are the respective weights on these points.

By performing the $\mathbf{c}_j\mathbf{x}_{jk}^*$ vector multiplications and the $\mathbf{A}_j\mathbf{x}_{jk}^*$ matrix multiplications, the following reformulated version of the overall problem can be obtained:

$$\text{Maximize} \qquad Z = 20\rho_{12} + 26\rho_{13} + 24\rho_{14} + 24\rho_{22} + 20\rho_{23},$$

subject to

$$\begin{aligned} 5\rho_{12} + 11\rho_{13} + 12\rho_{14} + \ 6\rho_{22} + 16\rho_{23} + x_{s1} &= 20 \\ 10\rho_{12} + 13\rho_{13} + 12\rho_{14} + 18\rho_{22} + 16\rho_{23} + x_{s2} &= 25 \\ \rho_{11} + \rho_{12} + \rho_{13} + \rho_{14} &= 1 \\ \rho_{21} + \rho_{22} + \rho_{23} &= 1 \end{aligned}$$

and

$$\begin{aligned} \rho_{1k} &\geq 0, \quad \text{for } k = 1, 2, 3, 4, \\ \rho_{2k} &\geq 0, \quad \text{for } k = 1, 2, 3, \\ x_{si} &\geq 0, \quad \text{for } i = 1, 2. \end{aligned}$$

However, we should emphasize that the complete reformulation normally is *not* constructed *explicitly;* rather, just parts of it are generated as needed during the progress of the revised simplex method.

To begin solving this problem, the initialization step selects x_{s1}, x_{s2}, ρ_{11}, and ρ_{21} to be the initial basic variables, so that

$$\mathbf{x}_B = \begin{bmatrix} x_{s1} \\ x_{s2} \\ \rho_{11} \\ \rho_{21} \end{bmatrix}.$$

Therefore, since $\mathbf{A}_1\mathbf{x}_{11}^* = 0$, $\mathbf{A}_2\mathbf{x}_{21}^* = 0$, $\mathbf{c}_1\mathbf{x}_{11}^* = 0$, and $\mathbf{c}_2\mathbf{x}_{21}^* = 0$, then

$$\mathbf{B} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \mathbf{B}^{-1}, \qquad \mathbf{x}_B = \mathbf{b}' = \begin{bmatrix} 20 \\ 25 \\ 1 \\ 1 \end{bmatrix}, \qquad \mathbf{c}_B = [0, 0, 0, 0]$$

for the initial BF solution.

To begin testing for optimality, let $j = 1$, and solve the linear programming problem

$$\text{Minimize} \qquad W_1 = (\mathbf{0} - \mathbf{c}_1)\mathbf{x}_1 + 0 = -4x_1 - 6x_2,$$

subject to

$$\mathbf{A}_3\mathbf{x}_1 \leq \mathbf{b}_1 \qquad \text{and} \qquad \mathbf{x}_1 \geq \mathbf{0},$$

so the feasible region is that shown in Fig. 23.3. Using Fig. 23.3 to solve graphically, the solution is

$$\mathbf{x}_1 = \begin{bmatrix} 2 \\ 3 \end{bmatrix} = \mathbf{x}_{13}^*,$$

so that $W_1^* = -26$.

Next let $j = 2$, and solve the problem

$$\text{Minimize} \qquad W_2 = (\mathbf{0} - \mathbf{c}_2)\mathbf{x}_2 + 0 = -8x_3 - 5x_4,$$

subject to

$$\mathbf{A}_4\mathbf{x}_2 \leq \mathbf{b}_2 \qquad \text{and} \qquad \mathbf{x}_2 \geq \mathbf{0},$$

so Fig. 23.4 shows this feasible region. Using Fig. 23.4, the solution is

$$\mathbf{x}_2 = \begin{bmatrix} 3 \\ 0 \end{bmatrix} = \mathbf{x}_{22}^*,$$

so $W_2^* = -24$. Finally, since none of the slack variables are nonbasic, no more coefficients in the current Eq. (0) need to be calculated. It can now be concluded that because both $W_1^* < 0$ and $W_2^* < 0$, the current BF solution is *not* optimal. Furthermore, since W_1^* is the smaller of these, ρ_{13} is the new entering basic variable.

For the revised simplex method to now determine the leaving basic variable, it is first necessary to calculate the column of $\mathbf{A}'$ giving the original coefficients of ρ_{13}. This column is

$$\mathbf{A}'_k = \begin{bmatrix} \mathbf{A}_1\mathbf{x}_{13}^* \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 11 \\ 13 \\ 1 \\ 0 \end{bmatrix}.$$

Proceeding in the usual way to calculate the current coefficients of ρ_{13} and the right-side column,

$$\mathbf{B}^{-1}\mathbf{A}'_k = \begin{bmatrix} 11 \\ 13 \\ 1 \\ 0 \end{bmatrix}, \qquad \mathbf{B}^{-1}\mathbf{b}' = \begin{bmatrix} 20 \\ 25 \\ 1 \\ 1 \end{bmatrix}.$$

Considering just the strictly positive coefficients, the *minimum ratio* of the right side to the coefficient is the $^1/_1$ in the third row, so that $r = 3$; that is, ρ_{11} is the new leaving basic variable. Thus, the new values of $\mathbf{x}_B$ and $\mathbf{c}_B$ are

$$\mathbf{x}_B = \begin{bmatrix} x_{s1} \\ x_{s2} \\ \rho_{13} \\ \rho_{21} \end{bmatrix}, \qquad \mathbf{c}_B = [0, 0, 26, 0].$$

To find the new value of $\mathbf{B}^{-1}$, set

$$\mathbf{E} = \begin{bmatrix} 1 & 0 & -11 & 0 \\ 0 & 1 & -13 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

so

$$\mathbf{B}_{\text{new}}^{-1} = \mathbf{E}\mathbf{B}_{\text{old}}^{-1} = \begin{bmatrix} 1 & 0 & -11 & 0 \\ 0 & 1 & -13 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

The stage is now set for again testing whether the current BF solution is optimal. In this case

$$W_1 = (\mathbf{0} - \mathbf{c}_1)\mathbf{x}_1 + 26 = -4x_1 - 6x_2 + 26,$$

so the minimum feasible solution from Fig. 23.3 is again

$$\mathbf{x}_1 = \begin{bmatrix} 2 \\ 3 \end{bmatrix} = \mathbf{x}_{13}^*,$$

with $W_1^* = 0$. Similarly,

$$W_2 = (\mathbf{0} - \mathbf{c}_2)\mathbf{x}_2 + 0 = -8x_3 - 5x_4,$$

so the minimizing solution from Fig. 23.4 is again

$$\mathbf{x}_2 = \begin{bmatrix} 3 \\ 0 \end{bmatrix} = \mathbf{x}_{22}^*,$$

with $W_2^* = -24$. Finally, there are no nonbasic slack variables to be considered. Since $W_2^* < 0$, the current solution is not optimal, and ρ_{22} is the new entering basic variable.

Proceeding with the revised simplex method,

$$\mathbf{A}_k' = \begin{bmatrix} \mathbf{A}_2\mathbf{x}_{22}^* \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 6 \\ 18 \\ 0 \\ 1 \end{bmatrix},$$

so

$$\mathbf{B}^{-1}\mathbf{A}_k' = \begin{bmatrix} 6 \\ 18 \\ 0 \\ 1 \end{bmatrix}, \qquad \mathbf{B}^{-1}\mathbf{b}' = \begin{bmatrix} 9 \\ 12 \\ 1 \\ 1 \end{bmatrix}.$$

Therefore, the minimum positive ratio is $\frac{12}{18}$ from the second row, so $r = 2$; that is, x_{s_2} is the new leaving basic variable. Thus

$$\mathbf{E} = \begin{bmatrix} 1 & -\frac{1}{3} & 0 & 0 \\ 0 & \frac{1}{18} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -\frac{1}{18} & 0 & 1 \end{bmatrix},$$

$$\mathbf{B}_{\text{new}}^{-1} = \mathbf{E}\mathbf{B}_{\text{old}}^{-1} = \begin{bmatrix} 1 & -\frac{1}{3} & -\frac{20}{3} & 0 \\ 0 & \frac{1}{18} & -\frac{13}{18} & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -\frac{1}{18} & \frac{13}{18} & 1 \end{bmatrix}, \qquad \mathbf{x}_B = \begin{bmatrix} x_{s1} \\ \rho_{22} \\ \rho_{13} \\ \rho_{21} \end{bmatrix},$$

and $\mathbf{c}_B = [0, 24, 26, 0]$.

Now test whether the new BF solution is optimal. Since

$$W_1 = \left([0, 24, 26, 0] \begin{bmatrix} 1 & -\frac{1}{3} \\ 0 & \frac{1}{18} \\ 0 & 0 \\ 0 & -\frac{1}{18} \end{bmatrix} \begin{bmatrix} 1 & 3 \\ 2 & 3 \end{bmatrix} - [4, 6] \right) \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + [0, 24, 26, 0] \begin{bmatrix} -\frac{20}{3} \\ -\frac{13}{18} \\ 1 \\ \frac{13}{18} \end{bmatrix}$$

$$= \left([0, \ \tfrac{4}{3}] \begin{bmatrix} 1 & 3 \\ 2 & 3 \end{bmatrix} - [4, 6] \right) \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \tfrac{26}{3}$$

$$= -\tfrac{4}{3}x_1 - 2x_2 + \tfrac{26}{3}.$$

Fig. 23.3 indicates that the minimum feasible solution is again

$$\mathbf{x}_1 = \begin{bmatrix} 2 \\ 3 \end{bmatrix} = \mathbf{x}_{13}^*,$$

so $W_1^* = \frac{2}{3}$. Similarly,

$$W_2 = \left([0, \tfrac{4}{3}]\begin{bmatrix} 2 & 4 \\ 6 & 4 \end{bmatrix} - [8, 5]\right)\begin{bmatrix} x_3 \\ x_4 \end{bmatrix} + 0$$

$$= 0x_3 + \tfrac{1}{3}x_4,$$

so the minimizing solution from Fig. 23.4 now is

$$\mathbf{x}_2 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \mathbf{x}_{21}^*,$$

and $W_2^* = 0$. Finally, $\mathbf{c}_B(\mathbf{B}^{-1})_{1;m_0} = [-, \frac{4}{3}]$. Therefore, since $W_1^* \geq 0$, $W_2^* \geq 0$, and $\mathbf{c}_B(\mathbf{B}^{-1})_{1;m_0} \geq \mathbf{0}$, the current BF solution is *optimal.* To identify this solution, set

$$\mathbf{x}_B = \begin{bmatrix} x_{s1} \\ \rho_{22} \\ \rho_{13} \\ \rho_{21} \end{bmatrix} = \mathbf{B}^{-1}\mathbf{b}' = \begin{bmatrix} 1 & -\frac{1}{3} & -\frac{20}{3} & 0 \\ 0 & \frac{1}{18} & -\frac{13}{18} & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -\frac{1}{18} & \frac{13}{18} & 1 \end{bmatrix}\begin{bmatrix} 20 \\ 25 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 5 \\ \frac{2}{3} \\ 1 \\ \frac{1}{3} \end{bmatrix},$$

so

$$\mathbf{x}_1 = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \sum_{k=1}^{4} \rho_{1k}\mathbf{x}_{1k}^* = \mathbf{x}_{12}^* = \begin{bmatrix} 2 \\ 3 \end{bmatrix},$$

$$\mathbf{x}_2 = \begin{bmatrix} x_3 \\ x_4 \end{bmatrix} = \sum_{k=1}^{3} \rho_{2k}\mathbf{x}_{2k}^* = \tfrac{1}{3}\begin{bmatrix} 0 \\ 0 \end{bmatrix} + \tfrac{2}{3}\begin{bmatrix} 3 \\ 0 \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \end{bmatrix}.$$

Thus, an optimal solution for this problem is $x_1 = 2$, $x_2 = 3$, $x_3 = 2$, $x_4 = 0$, with $Z = 42$.

23.4 MULTITIME PERIOD PROBLEMS

Any successful organization must plan ahead and take into account probable changes in its operating environment. For example, predicted future changes in sales because of seasonal variations or long-run trends in demand might affect how the firm should operate currently. Such situations frequently lead to the formulation of multitime period linear programming problems for planning several time periods (e.g., days, months, or years) into the future. Just as for multidivisional problems, multitime period problems are *almost* decomposable into separate subproblems, where each subproblem in this case is concerned with optimizing the operation of the organization during one of the time periods. However, some overall planning is required to coordinate the activities in the different time periods.

The resulting special structure for multitime period problems is shown in Table 23.8. Each approximately square block gives the coefficients of the constraints for one subproblem concerned with optimizing the operation of the organization during a particular time period considered by itself. Each oblong block then contains the coefficients of the **linking variables** for those activities that affect two or more time periods. For example, the linking variables may describe inventories that are retained at the end of one time period for use in some later time period, as we shall illustrate in the prototype example.

As with multidivisional problems, the multiplicity of subproblems often causes multitime period problems to have a very large number of constraints and variables, so again a method for exploiting the *almost decomposable* special structure of these problems is needed. Fortunately, the *same* method can be used for both types of problems! The idea is to reorder the variables in the multitime period problem to first list all the linking variables, as shown in Table 23.9, and then to construct its dual problem. This dual problem

TABLE 23.8 Constraint coefficients for multitime period problems

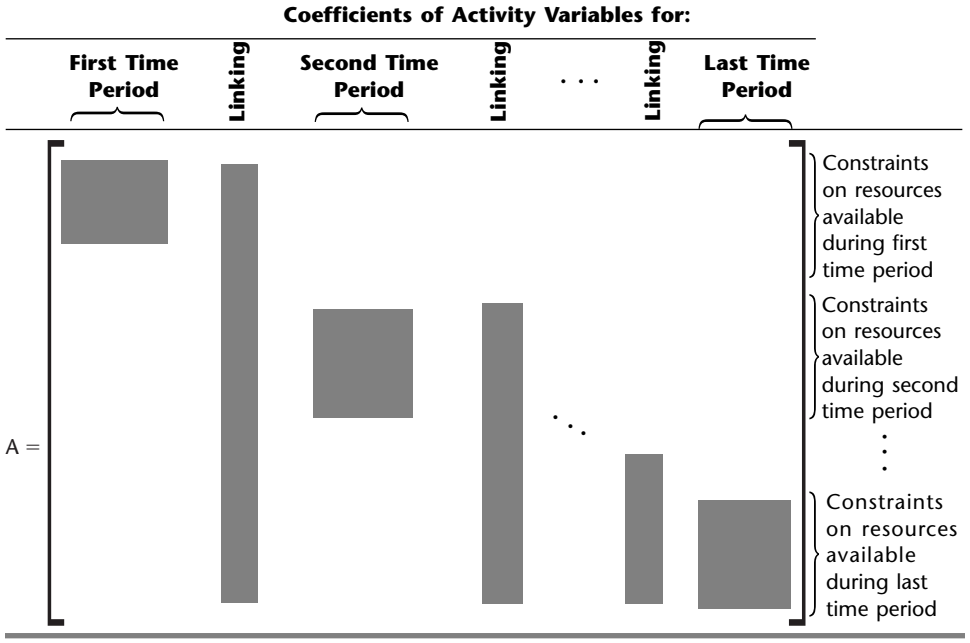
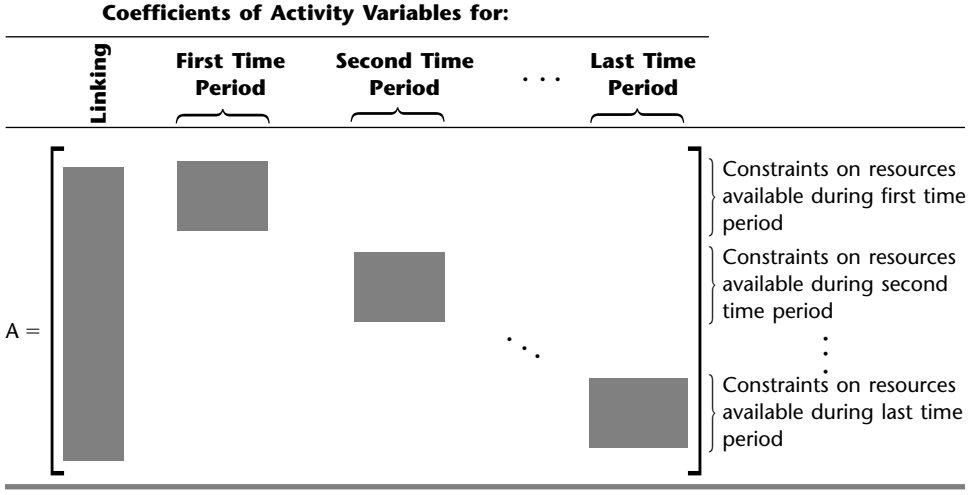


TABLE 23.9 Table of constraint coefficients for multitime period problems after reordering the variables



exactly fits the block angular structure shown in Table 23.4. (For this reason the special structure in Table 23.9 is referred to as the **dual angular structure.**) Therefore, the *decomposition principle* presented in the preceding section for multidivisional problems can be used to solve this dual problem. Since directly applying even this streamlined version of the simplex method to the dual problem automatically identifies an optimal solution for the primal problem as a by-product, this provides an efficient way of solving many large multitime period problems.

Prototype Example

The WOODSTOCK COMPANY operates a large warehouse that buys and sells lumber. Since the price of lumber changes during the different seasons of the year, the company sometimes builds up a large stock when prices are low and then stores the lumber for sale later at a higher price. The manager feels that there is considerable room for increasing profits by improving the scheduling of purchases and sales, so he has hired a team of operations research consultants to develop the most profitable schedule.

Since the company buys lumber in large quantities, its purchase price is slightly less than its selling price in each season. These prices are shown in Table 23.10, along with the maximum amount that can be sold during each season. The lumber would be purchased at the beginning of a season and sold throughout the season. If the lumber purchased is to be stored for sale in a later season, a handling cost of \$7 per 1,000 board feet is incurred, as well as a storage cost (including interest on capital tied up) of \$10 per 1,000 board feet for each season stored. A maximum of 2 million board feet can be stored in the warehouse at any one time. (This includes lumber purchased for sale in the same period.) Since lumber should not age too long before sale, the manager wants it all sold by the end of autumn (before the low winter prices go into effect).

The team of OR consultants concluded that this problem should be formulated as a linear programming problem of the multitime period type. Numbering the seasons (1 = winter, 2 = spring, 3 = summer, 4 = autumn) and letting x_i be the number of 1,000 board feet purchased in season i, y_i be the number sold in season i, and z_{ij} be the number stored in season i for sale in season j, this formulation is

$$\text{Maximize} \quad Z = -410x_1 + 425y_1 - 17z_{12} - 27z_{13} - 37z_{14} - 430x_2 + 440y_2 - 17z_{23} - 27z_{24} - 460x_3 + 465y_3 - 17z_{34} - 450x_4 + 455y_4,$$

subject to

$$\begin{aligned}
&x_1 - y_1 - z_{12} - z_{13} - z_{14} = 0\\
&x_1 \le 2000\\
&y_1 \le 1000\\
&z_{12} + x_2 - y_2 - z_{23} - z_{24} = 0\\
&z_{12} - y_2 \le 0\\
&z_{12} + z_{13} + z_{14} + x_2 \le 2000\\
&y_2 \le 1400\\
&z_{13} + z_{23} + x_3 - y_3 - z_{34} = 0\\
&z_{13} + z_{23} - y_3 \le 0\\
&z_{13} + z_{14} + z_{23} + z_{24} + x_3 \le 2000\\
&y_3 \le 2000\\
&z_{14} + z_{24} + z_{34} + x_4 - y_4 = 0\\
&y_4 \le 1600
\end{aligned}$$

TABLE 23.10 Price data for the Woodstock Company

Season	Purchase Price*	Selling Price*	Maximum Sales†
Winter	410	425	1,000
Spring	430	440	1,400
Summer	460	465	2,000
Autumn	450	455	1,600

*Prices are in dollars per thousand board feet.

†Sales are in thousand board feet.

TABLE 23.11 Table of constraint coefficients for the Woodstock Company multitime period problem after reordering the variables

Coefficient of:													
z_{12}	z_{13}	z_{14}	z_{23}	z_{24}	z_{34}	x_1	y_1	x_2	y_2	x_3	y_3	x_4	y_4

and

$$x_i \geq 0, \qquad y_i \geq 0, \qquad z_{ij} \geq 0, \qquad \text{for } i = 1, 2, 3, 4, \text{ and } j = 2, 3, 4.$$

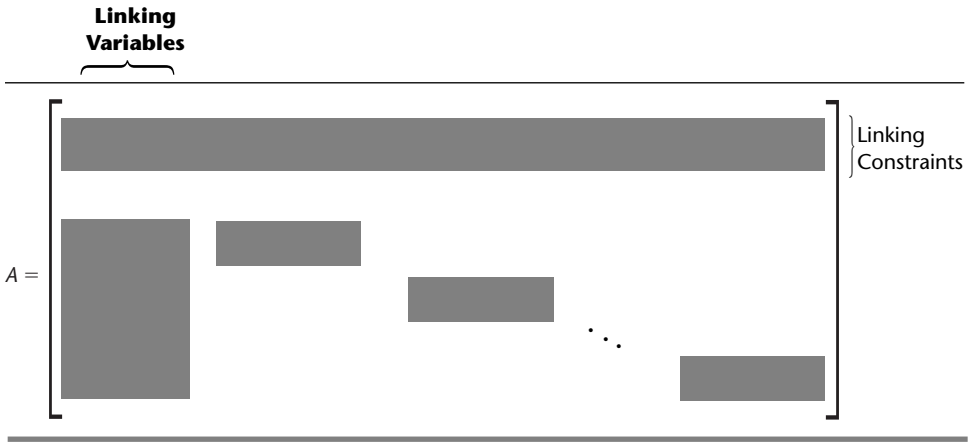
Thus, this formulation contains four subproblems, where the subproblem for season i is obtained by deleting all variables except x_i and y_i from the overall problem. The storage variables (the z_{ij}) then provide the *linking variables* that interrelate these four time periods. Therefore, after reordering the variables to first list these linking variables, the corresponding table of constraint coefficients has the form shown in Table 23.11, where *all* blanks are *zeros*. Since this form fits the dual angular structure given in Table 23.9, the streamlined solution procedure for this kind of special structure can be used to solve the problem (or much larger versions of it).

23.5 MULTIDIVISIONAL MULTITIME PERIOD PROBLEMS

You saw in the preceding two sections how decentralized decision making can lead to multidivisional problems and how a changing operating environment can lead to multitime period problems. We discussed these two situations separately to focus on their individual special structure. However, we should now emphasize that it is fairly common for problems to possess *both* characteristics simultaneously. For example, because costs and market prices change frequently in the food industry, the Good Foods Corp. might want to expand their multidivisional problem to consider the effect of such predicted changes several time periods into the future. This would allow the model to indicate how to most profitably stock up on materials when costs are low and store portions of the food products until prices are more favorable. Similarly, if the Woodstock Co. also owns several other warehouses, it might be advisable to expand their model to include and coordinate the activities of these divisions of their organization. (Also see Prob. 23.5-2 for another way in which the Woodstock Co. problem might expand to include the multidivisional structure.)

The combined special structure for such *multidivisional multitime period problems* is shown in Table 23.12. It contains many subproblems (the approximately square blocks), each of which is concerned with optimizing the operation of one division during one of the time periods considered in isolation. However, it also includes *both* linking constraints

TABLE 23.12 Constraint coefficients for multidivisional multitime period problems



and linking variables (the oblong blocks). The *linking constraints* coordinate the divisions by making them share the organizational resources available during one or more time periods. The linking variables coordinate the time periods by representing activities that affect the operation of a particular division (or possibly different divisions) during two or more time periods.

One way of exploiting the combined special structure of these problems is to apply an extended version of the decomposition principle for multidivisional problems. This involves treating everything but the linking constraints as one large subproblem and then using this decomposition principle to coordinate the solution for this subproblem with the master problem defined by the linking constraints. Since this large subproblem has the dual angular structure shown in Table 23.9, it would be solved by the special solution procedure for multitime period problems, which again involves using this decomposition principle.

Other procedures for exploiting this combined special structure also have been developed.[1] More experimentation is still needed to test the relative efficiency of the available procedures.

23.6 STOCHASTIC PROGRAMMING

One of the common problems in the practical application of linear programming is the difficulty of determining the proper values of the model parameters (the c_j, a_{ij}, and b_i). The true values of these parameters may not become known until after a solution has been chosen and implemented. This can sometimes be attributed solely to the inadequacy of the investigation. However, the values these parameters take on often are influenced by random events that are impossible to predict. In short, some or all of the model parameters may be *random variables*.

When these random variable parameters have relatively small variances, the standard approach is to perform sensitivity analysis as described in Chap. 6. However, if some of the parameters have relatively large variances, this approach is not very adequate. What

[1]For further information, see Chap. 5 of Selected Reference 7 at the end of this chapter.

is needed is a way of formulating the problem so that the optimization will directly take the uncertainty into account.

Some such approaches for *linear programming under uncertainty* have been developed. These formulations can be classified into two types, stochastic programming and chance-constrained programming, which are described in this and the next section, respectively. The main distinction between these types is that *stochastic programming* requires all constraints to hold with probability 1, whereas *chance-constrained programming* permits a small probability of violating any functional constraint. The former type was given its name because it is particularly applicable when the values of the decision variables are chosen at two or more different points in time (i.e., stochastically), although the latter type also can be adapted to this kind of multistage problem. The general approach for dealing with both types is to reformulate them as new equivalent linear programming problems where the certainty assumption *is* satisfied, and then solve by the simplex method. This clever reformulation for each type is the key to its practicality.

Focusing now on stochastic programming, we will introduce its main ideas only, largely through simple illustrative examples, rather than developing a complete formal description.

If some or all of the c_j are random variables, then

$$Z = \sum_{j=1}^{n} c_j x_j$$

also is a random variable for any given solution. Since it is meaningless to maximize a random variable, Z must be replaced by some deterministic function. There are many possible choices for this function, each of which may be very reasonable under certain circumstances. Perhaps the most natural choice, and certainly the most widely used, is the expected value of Z,

$$E(Z) = \sum_{j=1}^{n} E(c_j)x_j.$$

Similarly, the functional constraints

$$\sum_{j=1}^{n} a_{ij}x_j \le b_i, \qquad \text{for } i = 1, 2, \ldots, m$$

must be reinterpreted if any of the a_{ij} and b_i are random variables. One interpretation is that a solution is considered feasible only if it satisfies all the constraints for *all possible combinations* of the parameter values. This is the interpretation assumed in this section, although it is soon modified to allow certain random variable parameters to become known before values are assigned to certain x_j.

One danger with this strict interpretation of feasibility is that there may well not exist *any* solution that satisfies all the constraints for *every* possible combination of the parameter values. If so, a more liberal interpretation can be used, such as the one given in the next section.

The remainder of the section is devoted to elaborating on how stochastic programming implements its interpretation of feasibility for two categories of problems.

One-Stage Problems

A *one-stage problem* is one where the values for all the x_j must be chosen simultaneously (i.e., at one stage) before learning which value has been taken on by any of the random variable parameters. This is in contrast to the multistage problems considered later, where the decision making is done over two or more stages while observing the values taken on by some of the random variable parameters.

The formulation for one-stage problems is relatively straightforward. Consider first the case where a_{ij} and b_i that are random variables are mutually independent. Then each

of these a_{ij} and b_i with multiple possible values would be replaced by its most restrictive value for its constraint; i.e., functional constraint i becomes

$$\sum_{j=1}^{n} (\max a_{ij})x_j \leq \min b_i,$$

where max a_{ij} is the *largest* value that the random variable a_{ij} can take on and min b_i is the *smallest* value that the random variable b_i can take on. By replacing the random variables with these constants, the new constraint ensures that the original constraint will be satisfied for every possible combination of values for the random variable parameters. Furthermore, the new constraint satisfies the certainty assumption of linear programming discussed in Sec. 3.3, so the reformulated problem can be solved by the simplex method.

For example, consider the constraint,

$$a_{11}x_1 + a_{12}x_2 \leq b_1,$$

where a_{11}, a_{12}, and b_1 all are independent random variables having the following ranges of possible values:

$$1 \leq a_{11} \leq 2, \qquad 2 \leq a_{12} \leq 3, \qquad 4 \leq b_1 \leq 5.$$

To reformulate to satisfy the certainty assumption of linear programming, this constraint should be replaced by

$$2x_1 + 3x_2 \leq 4.$$

Reformulating a constraint in this manner is more restrictive than necessary if the random variable parameters are jointly dependent in a way that prevents the parameters from simultaneously achieving their most restrictive values. A case of special interest is where, at least as an approximation, the problem can be described as having a relatively small number of possible scenarios for how the problem will unfold over time, where each scenario provides certain fixed values for all the parameters. Which scenario will occur may depend on some exogenous factor, such as the state of the economy, or the market's reception to new products, or the extent of progress on new technological advances.

For this kind of situation, the original constraint with random variables would be replaced by a set of new constraints, where each new constraint would have the parameter values that correspond to one of the scenarios. For example, consider again the constraint,

$$a_{11}x_1 + a_{12}x_2 \leq b_1,$$

but suppose now that a_{11}, a_{12}, and b_1 each are random variables that have just the two possible values shown below:

$$a_{11} = 1 \text{ or } 2, \qquad a_{12} = 2 \text{ or } 3, \qquad b_1 = 4 \text{ or } 5.$$

Further suppose that there are just two scenarios, where each one dictates which of the two values each random variable will take on, as follows:

Scenario 1: $a_{11} = 1$, $a_{12} = 3$, $b_1 = 4$.
Scenario 2: $a_{11} = 2$, $a_{12} = 2$, $b_1 = 5$.

In this case, the original constraint with random variables would be replaced by the two new constraints,

$$x_1 + 3x_2 \leq 4$$
$$2x_1 + 2x_2 \leq 5.$$

This approach does have the drawback of increasing the number of functional constraints, which substantially increases the computation time for the simplex method. This drawback can become quite serious if a large number of scenarios need to be considered.

Multistage Problems

We now consider problems where the decisions on the values of the x_j are made at two or more points in time (stages). That is, some of the x_j are *first-stage variables,* others are *second-stage variables,* and so on. For example, this occurs when scheduling the production of some products over several time periods, where each x_j gives the production level for one of the products in one of the time periods.

Although the decisions are made in stages, they still need to be considered jointly in one model because the activities involved are consuming the same limited resources. However, the overall optimization makes the decisions for later stages conditional upon what happens at preceding stages, namely, the values taken on by some of the random variable parameters (typically the constraint coefficients for the variables associated with the preceding stages). Therefore, the stochastic programming approach enables adjusting the decisions for later stages based on unfolding circumstances.

The key idea for the stochastic programming formulation here is to replace each original decision variable beyond the first stage by a set of new decision variables, where each new decision variable represents the original decision under one of the possible circumstances that could prevail at the point.

To illustrate this approach, consider the problem,

$$\text{Maximize} \qquad Z = 3x_1 + 7x_2 + 11x_3,$$

subject to

$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 \leq 100$$

and

$$x_1 \geq 0, \qquad x_2 \geq 0, \qquad x_3 \geq 0,$$

where a_{11}, a_{12}, and a_{13} are independent random variables such that

$$a_{11} = \begin{cases} 1, & \text{with probability } \frac{1}{2} \\ 2, & \text{with probability } \frac{1}{2} \end{cases}$$

$$a_{12} = \begin{cases} 3, & \text{with probability } \frac{1}{2} \\ 4, & \text{with probability } \frac{1}{2} \end{cases}$$

$$a_{13} = \begin{cases} 5, & \text{with probability } \frac{1}{2} \\ 6, & \text{with probability } \frac{1}{2} \end{cases}$$

and where x_1, x_2, and x_3 are the decision variables for stages 1, 2, and 3, respectively. The value taken on by a_{11} will be known before the value of x_2 must be chosen, and the value taken on by a_{12} will be known before the value of x_3 must be chosen.

The stochastic programming formulation for this example replaces x_2 by the set of new decision variables,

x_{21} = value chosen for x_2 if $a_{11} = 1$
x_{22} = value chosen for x_2 if $a_{11} = 2$,

and then replaces x_3 by the set of new decision variables,

x_{31} = value chosen for x_3 if $a_{11} = 1$, $a_{12} = 3$
x_{32} = value chosen for x_3 if $a_{11} = 1$, $a_{12} = 4$
x_{33} = value chosen for x_3 if $a_{11} = 2$, $a_{12} = 3$
x_{34} = value chosen for x_3 if $a_{11} = 2$, $a_{12} = 4$.

The resulting reformulated problem is

$$\text{Maximize} \qquad E(Z) = 3x_1 + 7(\tfrac{1}{2})(x_{21} + x_{22}) + 11(\tfrac{1}{4})(x_{31} + x_{32} + x_{33} + x_{34}),$$

subject to

$$\begin{aligned} x_1 + 3x_{21} + 6x_{31} &\leq 100 \\ x_1 + 4x_{21} + 6x_{32} &\leq 100 \\ 2x_1 + 3x_{22} + 6x_{33} &\leq 100 \\ 2x_1 + 4x_{22} + 6x_{34} &\leq 100 \end{aligned}$$

and

$$x_1 \geq 0 \qquad \text{and all } x_{ij} \geq 0,$$

which is an ordinary linear programming problem that can be solved by the simplex method. Note that each of the four functional constraints represents one of the four possible combinations of values for a_{11} and a_{12}. The reason that all four constraints have $a_{13} = 6$ and there are not four additional constraints with $a_{13} = 5$ is that 6 is the most restrictive value of a_{13} for this last-stage parameter. In the objective function, the multipliers of $\frac{1}{2}$ and $\frac{1}{4}$ arise because these are the probabilities of the combinations of parameter values that result in using the respective variables (x_{21}, x_{22}, and then x_{31}, x_{32}, x_{33}, x_{34}) for determining the value of x_2 or x_3.

This example also illustrates how the stochastic programming approach greatly increases the size of the model to be solved, especially if the number of stages and the number of possible combinations of values for the random variable parameters are large. This problem is avoided by the approach described in the next section.

23.7 CHANCE-CONSTRAINED PROGRAMMING

Section 23.6 presented the stochastic programming approach to linear programming under uncertainty. Chance-constrained programming provides another way of dealing with this problem. This alternative approach may be used when it is highly desirable, but not absolutely essential, that the functional constraints hold.

When some or all of the parameters of the model are random variables, the stochastic programming formulation requires that all the functional constraints must hold for *all* possible combinations of values for these random variable parameters. By contrast, the chance-constrained programming formulation requires only that each constraint must hold for most of these combinations. More precisely, this formulation replaces the original linear programming constraints,

$$\sum_{j=1}^{n} a_{ij}x_j \leq b_i, \qquad \text{for } i = 1, 2, \ldots, m,$$

by

$$P\left\{\sum_{j=1}^{n} a_{ij}x_j \leq b_i\right\} \geq \alpha_i, \qquad \text{for } i = 1, 2, \ldots, m,$$

where the α_i are specified constants between zero and one (although they are normally chosen to be reasonably close to one). Therefore, a nonnegative solution $(x_1, x_2, \ldots, x_n)$ is considered to be feasible if and only if

$$P\left\{\sum_{j=1}^{n} a_{ij}x_j \leq b_i\right\} \geq \alpha_i, \qquad \text{for } i = 1, 2, \ldots, m.$$

Each complementary probability, $1 - \alpha_i$, represents the allowable risk that the random variables will take on values such that

$$\sum_{j=1}^{n} a_{ij}x_j > b_i.$$

Thus, the objective is to select the "best" nonnegative solution that "probably" will turn out to satisfy each of the original constraints when the random variables (the a_{ij}, b_i, and c_j) take on their values.

There are many possible expressions for the objective function when some of the c_j are random variables, and several of these have been explored elsewhere[1] in the context of chance-constrained programming. However, only the one assumed in the preceding section, namely, the expected value function, is considered here.

No procedure is now available for solving the general chance-constrained (linear) programming problem. However, certain important special cases are solvable. The one discussed here is where: (1) all the a_{ij} parameters are constants, so that only some or all of the c_j and b_i are random variables, (2) the probability distribution of the b_i is a known multivariate normal distribution, and (3) c_j is statistically independent of b_i ($j = 1, 2, \ldots, n$; $i = 1, 2, \ldots, m$).

As in the preceding section, it is initially assumed that all of the x_j must be determined before learning the value taken on by any of the random variables. Then, after the approach for this case is developed, the more general case where this assumption is dropped will be discussed.

One-Stage Problems

The chance-constrained programming problem considered here fits the linear programming model format except for the constraints,

$$P\left\{\sum_{j=1}^{n} a_{ij}x_j \le b_i\right\} \ge \alpha_i, \qquad \text{for } i = 1, 2, \ldots, m.$$

Therefore, the goal is to convert these constraints into legitimate linear programming constraints, so that the simplex method can be used to solve the problem. This can be done under the stated assumptions, as shown below.

To begin, notice that

$$P\left\{\sum_{j=1}^{n} a_{ij}x_j \le b_i\right\} = P\left\{\frac{\sum_{j=1}^{n} a_{ij}x_j - E(b_i)}{\sigma_{b_i}} \le \frac{b_i - E(b_i)}{\sigma_{b_i}}\right\},$$

where $E(b_i)$ and σ_{b_i} are the mean and standard deviation of b_i, respectively. Since b_i is assumed to have a normal distribution, $[b_i - E(b_i)]/\sigma_{b_i}$ must also be normal with mean zero and standard deviation one. In the table for the normal distribution given in Appendix 5, K_α is taken to be the constant such that

$$P\{Y \ge K_\alpha\} = \alpha,$$

where α is any given number between zero and one, and where Y is the random variable whose probability distribution is normal with mean zero and standard deviation one. This table gives K_α for various values of α. For example,

$$K_{0.90} = -1.28,\ K_{0.95} = -1.645, \text{ and } K_{0.99} = -2.33.$$

[1]A. Charnes and W. W. Cooper, "Deterministic Equivalents for Optimizing and Satisficing under Chance Constraints," *Operations Research,* **11:** 18–39 (1963).

Therefore, it now follows that

$$P\left\{K_{\alpha_i} \leq \frac{b_i - E(b_i)}{\sigma_{b_i}}\right\} = \alpha_i.$$

Note that this probability would be increased if K_{α_i} were replaced by a number $< K_{\alpha_i}$. Hence,

$$P\left\{\frac{\sum_{j=1}^{n} a_{ij}x_j - E(b_i)}{\sigma_{b_i}} \leq \frac{b_i - E(b_i)}{\sigma_{b_i}}\right\} \geq \alpha_i$$

for a given solution if and only if

$$\frac{\sum_{j=1}^{n} a_{ij}x_j - E(b_i)}{\sigma_{b_i}} \leq K_{\alpha_i}.$$

Rewriting both expressions in an equivalent form, the conclusion is that

$$P\left\{\sum_{j=1}^{n} a_{ij}x_j \leq b_i\right\} \geq \alpha_i$$

if and only if

$$\sum_{j=1}^{n} a_{ij}x_j \leq E(b_i) + K_{\alpha_i}\sigma_{b_i},$$

so that this probability constraint can be replaced by this linear programming constraint. The fact that these constraints are equivalent is illustrated by Fig. 23.5.

To summarize, the chance-constrained programming problem considered above can be reduced to the following equivalent linear programming problem.

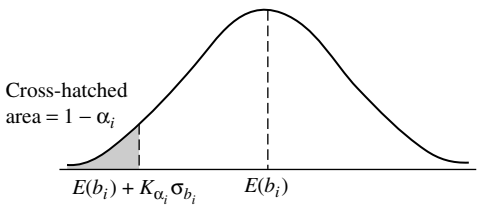
$$\text{Maximize} \qquad E(Z) = \sum_{j=1}^{n} E(c_j)x_j,$$

subject to

$$\sum_{j=1}^{n} a_{ij}x_j \leq E(b_i) + K_{\alpha_i}\sigma_{b_i}, \qquad \text{for } i = 1, 2, \ldots, m,$$

and

$$x_j \geq 0, \qquad \text{for } j = 1, 2, \ldots, n.$$



■ **FIGURE 23.5**
Probability density function of b_i.

Multistage Problems

We now will consider multistage problems such as discussed in the preceding section, where decisions beyond the first stage take into account the value taken on by certain random variable parameters at preceding stages. In our current context, we assume that some of the b_i become known before some of the x_j values must be chosen.

We need to formulate and solve problems of this type in such a way that the final decision on the x_j is partially based on the new information that has become available. The chance-constrained programming approach to this situation is to solve for each x_j as an explicit function of the b_i whose values become known before a value must be assigned to x_j. From a computational standpoint, it is convenient to deal with linear functions of the b_i, thereby leading to what are called *linear decision rules* for the x_j. In particular, let

$$x_j = \sum_{k=1}^{m} d_{jk} b_k + y_j, \qquad \text{for } j = 1, 2, \ldots, n,$$

where the d_{jk} are specified constants (where $d_{jk} = 0$ whenever the value taken on by b_k is not known before a value must be assigned to x_j), and where the y_j are decision variables.[1] (These equations are often written in matrix form as $\mathbf{x} = \mathbf{Db} + \mathbf{y}$.) The proper choice of the d_{jk} depends very much on the nature of the individual problem (if indeed it can be formulated reasonably in this way). An example is given later that illustrates how the d_{jk} are chosen.

Given the d_{jk}, it is only necessary to solve for the y_j. Then, when the time comes to assign a value to x_j, this value is obtained from the above equation. The details on how to solve for the y_j are given below.

The first step is to substitute

$$\left(\sum_{k=1}^{m} d_{jk} b_k + y_j \right) \qquad \text{for } x_j \qquad (\text{for } j = 1, 2, \ldots, n)$$

throughout the original chance-constrained programming model. The objective function becomes

$$E(Z) = E\left[\sum_{j=1}^{n} c_j \left(\sum_{k=1}^{n} d_{jk} b_k + y_j \right) \right]$$

$$= \sum_{j=1}^{n} \sum_{k=1}^{m} d_{jk} E(c_j) E(b_k) + \sum_{j=1}^{n} E(c_j) y_j.$$

Since

$$\sum_{j=1}^{n} \sum_{k=1}^{m} d_{jk} E(c_j) E(b_k)$$

is a constant, it can be dropped from the objective function, so that the new objective becomes

$$\text{Maximize} \qquad \sum_{j=1}^{n} E(c_j) y_j.$$

[1] Another common type of linear decision rule in chance-constrained programming is to let

$$x_j = \sum_{k=1}^{m} b_k d_{jk}, \qquad \text{for } j = 1, 2, \ldots, n,$$

where d_{jk} is a *decision variable* if b_k becomes known before a value must be assigned to x_j and is zero otherwise. This case is considered in Problem 23.7-2.

Since

$$\sum_{j=1}^{n} a_{ij}x_j = \sum_{j=1}^{n} a_{ij}\left(\sum_{k=1}^{m} d_{jk}b_k + y_j\right)$$

$$= \sum_{j=1}^{n}\sum_{k=1}^{m} a_{ij}d_{jk}b_k + \sum_{j=1}^{n} a_{ij}y_j,$$

the constraints,

$$P\left\{\sum_{j=1}^{n} a_{ij}x_j \le b_i\right\} \ge \alpha_i, \qquad \text{for } i = 1, 2, \ldots, m,$$

become

$$P\left\{\sum_{j=1}^{n} a_{ij}y_j \le b_i - \sum_{j=1}^{n}\sum_{k=1}^{m} a_{ij}d_{jk}b_k\right\} \ge \alpha_i, \qquad \text{for } i = 1, 2, \ldots, m.$$

The next step is to reduce these constraints to linear programming constraints. This is done just as before since the fundamental nature of the constraints has not been changed. Because

$$\left(b_i - \sum_{j=1}^{n}\sum_{k=1}^{m} a_{ij}d_{jk}b_k\right)$$

is a linear function of normal random variables, it must also be a normally distributed random variable. Let μ_i and σ_i denote the mean and standard deviation, respectively, of

$$\left(b_i - \sum_{j=1}^{n}\sum_{k=1}^{m} a_{ij}d_{jk}b_k\right).$$

Thus,

$$\mu_i = E(b_i) - \sum_{j=1}^{n}\sum_{k=1}^{m} a_{ij}d_{jk}E(b_k),$$

and, if the b_k are mutually independent,

$$\sigma_i^2 = \sum_{\substack{k=1\\k\neq i}}^{m}\left[\sum_{j=1}^{n} a_{ij}d_{jk}\right]^2 \sigma_{b_k}^2 + \left[1 - \sum_{j=1}^{n} a_{ij}d_{ji}\right]^2 \sigma_{bi}^2.$$

(Lacking independence, covariance terms would be included.) It then follows as before that these constraints are equivalent to the linear programming constraints,

$$\sum_{j=1}^{n} a_{ij}y_j \le \mu_i + K_{\alpha_i}\sigma_i, \qquad \text{for } j = 1, 2, \ldots, m.$$

It usually makes sense for the individual problem to add the restriction that

$$y_j \ge 0, \qquad \text{for } j = 1, 2, \ldots, n.$$

The model consisting of the new objective function and these constraints can then be solved by the simplex method.

To illustrate the way in which linear decision rules may arise, consider the problem of scheduling the production output for a given product over the next n time periods. Let x_j ($j = 1, 2, \ldots, n$) be the total number of units produced in time periods 1 through j, so that $(x_j - x_{j-1})$ is the output in period j. Thus, the x_j are the decision variables. Let S_j ($j = 1, 2, \ldots, n$) be the total number of units sold in time periods 1 through j. Assuming sales cannot be predicted exactly in advance, the S_j are random variables such that the value taken on by S_j becomes known at the end of period j. Assume that the S_j are normally distributed.

Suppose that the firm's management places a high priority on not alienating customers by a late delivery of their purchases. Hence, assuming no initial inventory, the x_j should be chosen such that it is almost certain that $x_j \geq S_j$. Therefore, one set of constraints that should be included in the mathematical model is

$$P\{x_j \geq S_j) \geq \alpha_j, \qquad \text{for } j = 1, 2, \ldots, n,$$

where the α_j are selected numbers close to one.

However, rather than solving for the x_j directly at the outset, the problem should be solved in such a way that the information on cumulative sales can be used as it becomes available. Suppose that the final decision on x_j need not be made until the beginning of period j. It would be highly desirable to take into account the value taken on by S_{j-1} before assigning a value to x_j. Therefore, let

$$x_j = S_{j-1} + y_j, \qquad \text{for } j = 1, 2, \ldots, n \text{ (where } S_0 = 0),$$

and then solve only for the y_j at the outset.

To express this example in the notation used earlier, the constraints should be written as

$$P\{-x_i \leq -S_i) \geq \alpha_i, \qquad \text{for } i = 1, 2, \ldots, m \ (m = n),$$

so that $b_i = -S_i$. Hence,

$$x_j = \sum_{k=1}^{m} d_{jk} b_k + y_j = -b_{j-1} + y_j,$$

so that $d_{j(j-1)} = -1$ and $d_{jk} = 0$ for $k \neq j - 1$. Since y_j is just the number of units of the product that is available for immediate delivery in period j, it is natural to impose the additional restriction that $y_j \geq 0$ for $j = 1, 2, \ldots, n$. Therefore, assuming that the remainder of the model also fits the linear programming format, this particular problem can be formulated and solved by the general procedure described in this section.

23.8 CONCLUSIONS

The linear programming model encompasses a wide variety of specific types of problems. The general simplex method is a powerful algorithm that can solve surprisingly large versions of any of these problems. However, some of these problem types have such simple formulations that they can be solved much more efficiently by *streamlined* versions of the simplex method that exploit their *special structure.* These streamlined versions can cut down tremendously on the computer time required for large problems, and they sometimes make it computationally feasible to solve huge problems. Of the problems considered in this chapter, this is particularly true for transshipment problems and problems with many upper-bound or GUB constraints. For general multidivisional problems, multitime period problems, or combinations of the two, the setup times are sufficiently large for their streamlined procedures that they should be used selectively only on large problems.

Stochastic programming and chance-constrained programming provide useful ways of dealing with linear programming problems where the certainty assumption is so badly violated that some or all of the model parameters must be treated explicitly as random variables.

Much research continues to be devoted to developing streamlined solution procedures for special types of linear programming problems, including some not discussed here. At the same time there is widespread interest in applying linear programming to optimize the operation of complicated large-scale systems, including social systems. The resulting formulations usually have special structures that can be exploited. Recognizing and exploiting special structures has become a very important factor in the successful application of linear programming.

SELECTED REFERENCES

1. Bazaraa, M. S., J. J. Jarvis, and H. D. Sherali: *Linear Programming and Network Flows,* 2d ed., Wiley, New York, 1990.
2. Bertsimas, D., and J. N. Tsitsiklis: *Introduction to Linear Optimization,* Athena Scientific, Belmont, MA, 1997.
3. Birge, J. R.: "Decomposition and Partitioning Methods for Multi-stage Stochastic Linear Programs," *Operations Research,* **33:** 989–1007, 1985.
4. Dantzig, G. B., and M. N. Thapa: *Linear Programming 2: Theory and Extensions,* Springer, New York, 2003.
5. Ermoliev, Y., and R. J. -B. Wets: *Numerical Techniques for Stochastic Optimization,* Springer-Verlag, New York, 1988.
6. Geoffrion, A. M.: "Elements of Large-Scale Mathematical Programming," *Management Science,* **16:** 652–691, 1970.
7. Ho, J. K.: "Recent Advances in the Decomposition Approach to Linear Programming," *Mathematical Programming Study 31* (a publication of the Mathematical Programming Society), pp. 119–128, 1987.
8. Infanger, G.: *Planning under Uncertainty,* Boyd and Fraser, Danvers, MA, 1994.
9. Lasdon, L. S.: *Optimization Theory for Large Systems,* Macmillan, New York, 1970.
10. Nemhauser, G. L.: "The Age of Optimization: Solving Large-Scale Real-World Problems," *Operations Research,* **42:** 5–13, 1994.
11. Rockafellar, R. T., and R. J. -B. Wets: "Scenario and Policy Aggregation in Optimization under Uncertainty," *Mathematics of Operations Research,* **16:** 119–147, 1991.
12. Rockafellar, R. T., and R. J. -B. Wets: *Variational Analysis,* Springer-Verlag, Berlin, 1998.

PROBLEMS

To the left of each of the following problems (or their parts), we have inserted a C whenever you should use the computer with any of the software options available to you (or as instructed by your instructor) to solve the problem.

23.1-1. Suppose that the air freight charge per ton between seven particular locations is given by the following table (except where no direct air freight service is available):

Location	1	2	3	4	5	6	7
1	—	21	50	62	93	77	—
2	21	—	17	54	67	—	48
3	50	17	—	60	98	67	25
4	62	54	60	—	27	—	38
5	93	67	98	27	—	47	42
6	77	—	67	—	47	—	5
7	—	48	25	38	42	35	—

A certain corporation must ship a certain perishable commodity from locations 1–3 to locations 4–7. A total of 70, 80, and 50 tons of this commodity is to be sent from locations 1, 2, and 3, respectively. A total of 30, 60, 50, and 60 tons is to be sent to locations 4, 5, 6, and 7, respectively. Shipments can be sent through intermediate locations at a cost equal to the sum of the costs for each of the legs of the journey. The problem is to determine the shipping plan that minimizes the total freight cost.

(a) Describe how this problem fits into the format of the general transshipment problem.

(b) Reformulate this problem as an equivalent transportation problem by constructing the appropriate parameter table.

(c) Use the northwest corner rule to obtain an initial BF solution for the problem formulated in part (*b*). Describe the corresponding shipping pattern.

C **(d)** Use the computer to obtain an optimal solution for the problem formulated in part (*b*). Describe the corresponding optimal shipping pattern.

23.1-2. Consider the airline company problem presented in Prob. 9.3-2.

(a) Describe how this problem can be fitted into the format of the transshipment problem.

(b) Reformulate this problem as an equivalent transportation problem by constructing the appropriate parameter table.

(c) Use Vogel's approximation method to obtain an initial BF solution for the problem formulated in part (*b*).

(d) Use the transportation simplex method by hand to obtain an optimal solution for the problem formulated in part (*b*).

23.1-3. A student about to enter college away from home has decided that she will need an automobile during the next four years. Since funds are going to be very limited, she wants to do this in the cheapest possible way. However, considering both the initial purchase price and the operating maintenance costs, it is not clear whether she should purchase a very old car or just a moderately old car. Furthermore, it is not clear whether she should plan to trade in her car at least once during the four years, before the costs become to high.

The relevant data *each* time she purchases a car are as follows:

	Purchase Price	Operating and Maintenance Costs for Ownership Year				Trade-in Value at End of Ownership Year			
		1	2	3	4	1	2	3	4
Very old car	\$1,200	\$1,900	\$2,200	\$2,500	\$2,800	\$ 700	\$ 500	\$ 400	\$ 300
Moderately old car	\$4,500	\$1,000	\$1,300	\$1,700	\$2,300	\$2,500	\$1,800	\$1,300	\$1,000

If the student trades in a car during the next four years, she would do it at the end of a year (during the summer) on another car of one of these two kinds. She definitely plans to trade in her car at the end of the four years on a much newer model. However, she needs to determine which plan for purchasing and (perhaps) trading in cars during the four years would minimize the *total* net cost for the four years.

(a) Describe how this problem can be fitted into the format of the transshipment problem.

(b) Reformulate this problem as an equivalent transportation problem by constructing the appropriate parameter table.

C **(c)** Use the computer to obtain an optimal solution for the problem formulated in part (*b*).

23.1-4. Without using x_{ii} variables to introduce fictional shipments from a location to itself, formulate the linear programming model for the general transshipment problem described at the end of Sec. 23.1. Identify the special structure of this model by constructing its table of constraint coefficients (similar to Table 23.1) that shows the location and values of the nonzero coefficients.

23.2-1. Consider the following linear programming problem.

Maximize $Z = 2x_1 + 4x_2 + 3x_3 + 2x_4 - 5x_5 + 3x_6$,

subject to

$$\begin{aligned} 3x_1 + 2x_2 + 3x_3 &\le 30 \\ 2x_5 - x_6 &\le 20 \\ 5x_1 - 2x_2 + 3x_3 + 4x_4 + 2x_5 + x_6 &\le 20 \\ 3 \le x_4 &\le 15 \\ 2x_5 + 3x_6 &\le 40 \\ 5x_1 - x_3 &\le 30 \\ 2x_1 + 4x_2 + 2x_4 + 3x_6 &\le 60 \\ -x_1 + 2x_2 + x_3 &\ge 20 \end{aligned}$$

and

$$x_j \ge 0, \qquad \text{for } j = 1, 2, \ldots, 6.$$

(a) Rewrite this problem in a form that demonstrates that it possesses the special structure for multidivisional problems. Identify the variables and constraints for the master problem and each subproblem.

(b) Construct the corresponding table of constraint coefficients having the block angular structure shown in Table 23.4. (Include only nonzero coefficients, and draw a box around each block of these coefficients to emphasize this structure.)

23.2-2. Consider the following table of constraint coefficients for a linear programming problem:

	Coefficient of:						
Constraint	x_1	x_2	x_3	x_4	x_5	x_6	x_7
1		1			1		1
2				1			
3	4	3	−2	2	4		1
4			2			4	
5	1			1			
6		5	3		1	−2	4
7						1	
8		2			1		3
9	2			4			

(a) Show how this table can be converted into the block angular structure for multidivisional linear programming as shown in Table 23.4 (with three subproblems in this case) by reordering the variables and constraints appropriately.

(b) Identify the upper-bound constraints and GUB constraints for this problem.

23.2-3. A corporation has two divisions (the Eastern Division and the Western Division) that operate semiautonomously, with each developing and marketing its own products. However, to coordinate their product lines and to promote efficiency, the divisions compete at the corporate level for investment funds for new product development projects. In particular, each division submits its proposals to corporate headquarters in September for new major projects to be undertaken the following year, and available funds are then allocated in such a way as to maximize the estimated total net discounted profits that will eventually result from the projects.

For the upcoming year, each division is proposing three new major projects. Each project can be undertaken at any level, where the estimated net discounted profit would be *proportional* to the level. The relevant data on the projects are summarized as follows:

	Eastern Division Project			Western Division Project		
	1	**2**	**3**	**1**	**2**	**3**
Level	x_1	x_2	x_3	x_4	x_5	x_6
Required investment (in millions of dollars)	$16x_1$	$7x_2$	$13x_3$	$8x_4$	$20x_5$	$10x_6$
Net profitability	$7x_1$	$3x_2$	$5x_3$	$4x_4$	$7x_5$	$5x_6$
Facility restriction	$10x_1 + 3x_2 +$		$7x_3 \leq 50$	$6x_4 + 13x_5 +$		$9x_6 \leq 45$
Labor restriction	$4x_1 + 2x_2 +$		$5x_3 \leq 30$	$3x_4 + 8x_5 +$		$2x_6 \leq 25$

A total of \$150,000,000 is budgeted for investment in these projects.

(a) Formulate this problem as a multidivisional linear programming problem.

(b) Construct the corresponding table of constraint coefficients having the block angular structure shown in Table 23.4.

23.3-1. Use the decomposition principle to solve the Wyndor Glass Co. problem presented in Sec. 3.1.

23.3-2. Consider the following multidivisional problem:

Maximize $\quad Z = 10x_1 + 5x_2 + 8x_3 + 7x_4,$

subject to

$$\begin{aligned} 6x_1 + 5x_2 + 4x_3 + 6x_4 &\leq 40 \\ 3x_1 + x_2 &\leq 15 \\ x_1 + x_2 &\leq 10 \\ x_3 + 2x_4 &\leq 10 \\ 2x_3 + x_4 &\leq 10 \end{aligned}$$

and

$$x_j \geq 0, \quad \text{for} \quad j = 1, 2, 3, 4.$$

(a) Explicitly construct the complete *reformulated* version of this problem in terms of the ρ_{jk} decision variables that would be generated (as needed) and used by the decomposition principle.

(b) Use the decomposition principle to solve this problem.

23.3-3. Using the decomposition principle, *begin* solving the Good Foods Corp. multidivisional problem presented in Sec. 23.2 by executing the first *two* iterations.

23.4-1. Consider the following table of constraint coefficients for a linear programming problem:

Constraint	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
1	3	1								
2	1	2	−1							
3				1	5					
4			1	2	−1	−1	−1			
5					1					
6			1			1	1	1	3	2
7								2	−1	1

Show how this table can be converted into the dual angular structure for multitime period linear programming shown in Table 23.9 (with three time periods in this case) by reordering the variables and constraints appropriately.

23.4-2. Consider the Wyndor Glass Co. problem described in Sec. 3.1 (see Table 3.1). Suppose that decisions have been made to discontinue additional products in the future and to initiate other new products. Therefore, for the two products being analyzed, the number of hours of production time available per week in each of the three plants will be different than shown in Table 3.1 after the first year. Furthermore, the profit per batch (exclusive of storage costs) that can be realized from the sale of these two products will vary from year to year as market conditions change. Therefore, it may be worthwhile to *store* some of the units produced in 1 year for sale in a later year. The storage costs involved would be approximately \$2,000 per batch for either product.

The relevant data for the next three years are summarized next.

		Hours/Week Available in Year		
		1	2	3
Plant	1	4	6	3
	2	12	12	10
	3	18	24	15
Profit per batch, Product 1		$3,000	$4,000	$5,000
Profit per batch, Product 2		$5,000	$4,000	$8,000

The production time per batch used by each product remains the same for each year as shown in Table 3.1. The objective is to determine how much of each product to produce in each year and what portion to store for sale in each subsequent year to maximize the total profit over the three years.

(a) Formulate this problem as a multitime period linear programming problem.

(b) Construct the corresponding table of constraint coefficients having the dual angular structure shown in Table 23.9.

23.5-1. Consider the following table of constraint coefficients for a linear programming problem.

Constraint	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
1	2			3				1		
2		1	1				2	2		
3	5	−1	2	−1	−1		−3			4
4						1		−1		
5		−1			2			−2	5	3
6	1			1						
7	2	1		3		2		1	−1	
8		−1	2				1	−1		
9					1				2	1
10		−1			4				1	5

Show how this table can be converted into the form for multidivisional multitime period problems shown in Table 23.12 (with two linking constraints, two linking variables, and four subproblems in this case) by reordering the variables and constraints appropriately.

23.5-2. Consider the Woodstock Company multitime period problem described in Sec. 23.4 (see Table 23.10). Suppose that the company has decided to expand its operation to also buy, store, and sell *plywood* in this warehouse. For the upcoming year, the relevant data for *raw lumber* are still as given in Sec. 23.4. The corresponding price data for plywood are as follows:

Season	Purchase Price*	Selling Price*	Maximum Sales†
Winter	680	705	800
Spring	715	730	1,200
Summer	760	770	1,500
Autumn	740	750	100

*Prices are in dollars per 1,000 board feet.

†Sales are in 1,000 board feet.

For plywood stored for sale in a later season, the handling cost is $6 per 1,000 board feet, and the storage cost is $18 per 1,000 board feet. The storage capacity of 2 million board feet now applies to the *total* for raw lumber and plywood. Everything should still be sold by the end of autumn.

The objective now is to determine the most profitable schedule for buying and selling raw lumber *and* plywood.

(a) Formulate this problem as a multidivisional multitime period linear programming problem.

(b) Construct the corresponding table of constraint coefficients having the form shown in Table 23.12.

23.6-1. Consider the following problem.

$$\text{Maximize} \qquad Z = 20x_1 + 30x_2 + 25x_3,$$

subject to

$$\begin{aligned} 3x_1 + 2x_2 + x_3 &\le b_1 \\ 2x_1 + 4x_2 + 2x_3 &\le b_2 \\ x_1 + 3x_2 + 5x_3 &\le b_3 \end{aligned}$$

and

$$x_j \ge 0, \qquad \text{for } j = 1, 2, 3,$$

where b_1, b_2, and b_3 are random variables. Assume that the probability distribution of each of these random variables is such that it can take on any one of three possible values. These values are (29, 30, 31) for b_1, (48, 50, 52) for b_2, and (57, 60, 63) for b_3. In each case, the probability of the middle value is 1/2, whereas each of the other two values has a probability of 1/4. The random variables are statistically independent. Suppose that the constraints are required to hold with probability 1.

(a) Reformulate this problem as an equivalent ordinary linear programming problem.

(b) Suppose that the value taken on by b_1 will be known when a value must be assigned to x_2, and both b_1 and b_2 will be known when x_3 must be specified. Use the stochastic programming approach to formulate an equivalent ordinary linear programming problem that maximizes $E(Z)$ while taking this information into account.

23.7-1. Reconsider Prob. 23.6-1. Suppose, after further analysis, it is decided that b_1, b_2, and b_3 each actually has a normal distribution, with a mean and standard deviation of (30, 1), (50, 2), and (60, 3), respectively. Therefore, a chance-constrained programming approach is to be used instead, where the first, second, and third constraints are required to hold with probability 0.975, 0.95, and 0.90, respectively.

(a) Consider the solution, $(x_1, x_2, x_3) = (2\frac{1}{3}, 7\frac{1}{3}, 6\frac{1}{3})$. What are the probabilities that the respective original constraints will be satisfied by this solution? Is this solution feasible? What is the probability that *all* the original constraints will be satisfied by this solution?

(b) Reformulate this chance-constrained programming problem as an equivalent ordinary linear programming problem.

(c) Suppose that [as in part (*b*) of Prob. 23.6-1] the value taken on by b_1 will be known when a value must be assigned to x_2, and

both b_1 and b_2 will be known when x_3 must be specified. Use the linear decision rules,

$$x_2 = \tfrac{1}{4}b_1 - y_2,$$

$$x_3 = \tfrac{1}{2}b_1 + \tfrac{1}{2}b_2 - y_3,$$

in order to formulate an equivalent ordinary linear programming problem that maximizes $E(Z)$ while taking this information into account.

23.7-2. Consider the chance-constrained programming constraint,

$$P\left\{\sum_{i=1}^{n} a_{ij}x_j \le b_i\right\} \ge \alpha_i.$$

(a) Suppose that, in addition to b_i, the a_{ij} also are (independent) random variables whose probability distributions are normal with known mean $E(a_{ij})$ and variance $\text{Var}(a_{ij})$. Convert this constraint into an equivalent deterministic nonlinear constraint.

(b) Suppose that the x_j are expressed as linear decision rules of the form,

$$x_j = \sum_{k=1}^{m} b_k\, d_{jk}, \qquad \text{for } j = 1, 2, \ldots, n,$$

where each d_{jk} is a *decision variable* if the value taken on by b_k will be known when a value must be assigned to x_j, and is zero otherwise. Assume that the b_k are independent random variables with known normal distributions, and that the a_{ij} are constants. Convert this constraint into an equivalent constraint of the form obtained in part (*a*).