giving $\mathbf{y} = \left(\frac{1}{2}, \frac{1}{3}, \frac{1}{6}, 0\right)$. The knapsack problem

$$z^* = \max \quad \frac{1}{2}a_1 + \frac{1}{3}a_2 + \frac{1}{6}a_3 + 0a_4$$

$$\text{s.t.}$$

$$38a_1 + 32a_2 + 27a_3 + 18a_4 \leq 100$$

$$a_k \geq 0, \text{ integer}$$

gives the solution $\mathbf{a} = (2, 0, 0, 0)$ with $z^* = 1$. Hence, we have the optimal basis. Solving for $\mathbf{x}_B$ gives $(7, 10, 30, 20)$, implying that we produce 7 raws cut by pattern $(2, 0, 0, 0)$, 10 raws of pattern $(0, 3, 0, 0)$, 30 raws of pattern $(1, 1, 1, 0)$, and 20 raws of pattern $(2, 0, 0, 1)$, totaling 67 raws.

Note that the final solution was integral. What is interesting about the cutting stock problem is that quite often, for no theoretical reason, the solution is integral. Of course, this does not always happen. However, when it does not happen, typically there is an integer solution with the same optimal value. However, getting such a solution is often difficult.

Cutting stock problems are not the only applications of column generation. For example, Degraeve and Schrage [28] use column generation to determine the production scheduling system for the curing operations at Bridgestone/Firestone Off-The-Road, which manufactures large tires for off-road machines. In this case, there are multiple constraints that determine a feasible tire mold formation, not just the one constraint we used in our example. Using real data from the manufacturing plant, less than 1000 of the millions of possible columns were generated during the algorithm.

11.4 DANTZIG–WOLFE DECOMPOSITION

In certain applications of linear programming, the constraints of the problem can be grouped into two classes: "easy" constraints and "hard" constraints. For example, the multicommodity flow problems we saw in Chapters 2 and 4 are such a case where the "hard" constraints correspond to the bounds on the total amount of flow on each arc and the "easy" constraints correspond to the flow constraints for each commodity. In general, the names "hard" and "easy" are misnomers; typically the "hard" constraints are the complicating ones, while the "easy" constraints would be those that, without the "hard" ones in the problem, would allow the problem to be efficiently solved.

When these applications have large numbers of variables and constraints, it is often useful to decompose our problem into one where we work on, at any given time, only the "easy" constraints or only the "hard" constraints. In this section, we discuss an approach for solving linear programs through such a decomposition method, known as **Dantzig–Wolfe Decomposition**. It is a form of column generation, where the columns are generated by solving the "easy" constraints along with some objective function.

Throughout this section, we will assume that our linear program can be written in the form

$$\begin{aligned} \max \quad & \mathbf{c}^T\mathbf{x} \\ \text{s.t.} \quad & \\ & A_H\mathbf{x} = \mathbf{b}_H \\ & A_E\mathbf{x} = \mathbf{b}_E \\ & \mathbf{x} \geq \mathbf{0}, \end{aligned} \tag{11.9}$$

where A_H (A_E) is the constraint matrix for the hard (easy) constraints.

■ EXAMPLE 11.12

We shall illustrate this approach with the following example problem throughout this section:

$$\begin{aligned} \max \quad & 8x_1 + 9x_2 + 7x_3 + 10x_4 \\ \text{s.t.} \quad & \\ & 2x_1 + x_2 + 4x_3 + 3x_4 = 20 \\ & x_1 + x_2 + x_3 + x_4 = 10 \\ & 3x_1 + x_2 \leq 6 \\ & 2x_3 + x_4 \leq 8 \\ & x_3 + x_4 \leq 6 \\ & x_1, x_2, x_3, x_4 \geq 0. \end{aligned} \tag{11.10}$$

The last three constraints will be considered the easy constraints and the first two the hard constraints, since if we remove the first two constraints we get the decoupled problems

$$\begin{aligned} \max \quad & 8x_1 + 9x_2 \\ \text{s.t.} \quad & \\ & 3x_1 + x_2 \leq 6 \\ & x_1, x_2 \geq 0 \end{aligned}$$

and

$$\begin{aligned} \max \quad & 7x_3 + 10x_4 \\ \text{s.t.} \quad & \\ & 2x_3 + x_4 \leq 8 \\ & x_3 + x_4 \leq 6 \\ & x_3, x_4 \geq 0, \end{aligned}$$

which can both be easily solved using graphical methods (Figures 11.4 and 11.5). After adding slack variables to each constraint, we would have

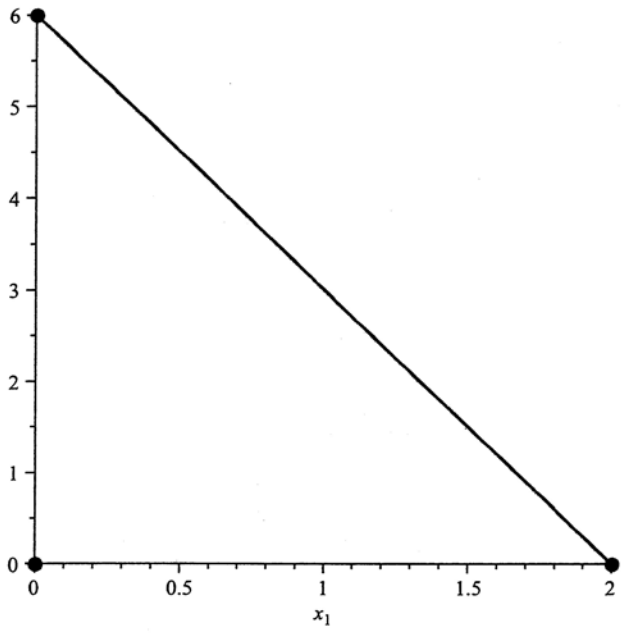

FIGURE 11.4 Feasible region for (x_1, x_2) variables.

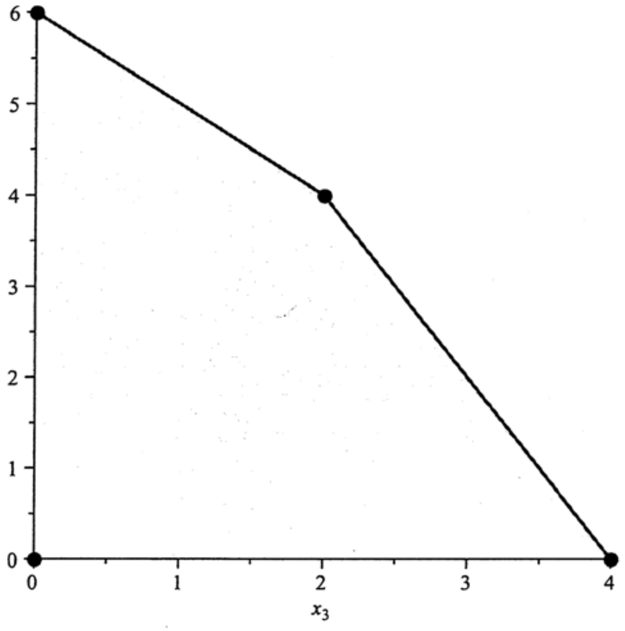

FIGURE 11.5 Feasible region for (x_3, x_4) variables.

$$A_H = \begin{bmatrix} 2 & 1 & 4 & 3 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 \end{bmatrix}$$

$$A_E = \begin{bmatrix} 3 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 2 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}.$$

Consider the feasible region for the "easy" constraints

$$X = \left\{\mathbf{x} : A_E\mathbf{x} = \mathbf{b}_E, \mathbf{x} \geq \mathbf{0}\right\} \tag{11.11}$$

and assume that X is bounded (this condition can be relaxed). By Theorem 7.4, if we have the set $V = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_K\}$ of extreme points of X, then every solution $\mathbf{x} \in X$ can be written as a convex combination of the extreme points in V, that is,

$$\begin{aligned} \mathbf{x} &= \sum_{j=1}^{K} \alpha_j \mathbf{v}_j \\ 1 &= \sum_{j=1}^{K} \alpha_j \\ \alpha_j &\geq 0. \end{aligned} \tag{11.12}$$

If we know every extreme point of X, then we can simply replace the "easy constraints" in (11.9) by (11.12) to get

$$\begin{aligned} \max \quad & \sum_{j=1}^{K} \left(\mathbf{c}^T\mathbf{v}_j\right)\alpha_j \\ \text{s.t.} \quad & \\ & \sum_{j=1}^{K} \left(A_H\mathbf{v}_j\right)\alpha_j = \mathbf{b}_H \\ & \sum_{j=1}^{K} \alpha_j = 1 \\ & \alpha_j \geq 0. \end{aligned} \tag{11.13}$$

The variables for (11.13) are the α_j's, and once we've solved (11.13), we can get the optimal solution to (11.9) by using (11.12).

■ EXAMPLE 11.13

Consider the easy constraints for (11.10). By examining the feasible region given in Figures 11.4 and 11.5 and adding slack variables, we can easily see that there are 12 total extreme points of X:

$$V = \left\{ (x_1, x_2, x_3, x_4, s_1, s_2, s_3) : \begin{array}{rl} \mathbf{v}_1 & = (0,0,0,0,6,8,6) \\ \mathbf{v}_2 & = (0,0,0,6,6,2,0) \\ \mathbf{v}_3 & = (0,0,2,4,6,0,0) \\ \mathbf{v}_4 & = (0,0,4,0,6,0,2) \\ \mathbf{v}_5 & = (0,6,0,0,0,8,6) \\ \mathbf{v}_6 & = (0,6,0,6,0,2,0) \\ \mathbf{v}_7 & = (0,6,2,4,0,0,0) \\ \mathbf{v}_8 & = (0,6,4,0,0,0,2) \\ \mathbf{v}_9 & = (2,0,0,0,0,8,6) \\ \mathbf{v}_{10} & = (2,0,0,6,0,2,0) \\ \mathbf{v}_{11} & = (2,0,2,4,0,0,0) \\ \mathbf{v}_{12} & = (2,0,4,0,0,0,2) \end{array} \right\}.$$

Thus, we would need to compute $\mathbf{c}^T\mathbf{v}_j$ and $A_H\mathbf{v}_j$ for each extreme point, which can be done as

$$\widehat{\mathbf{c}} = \begin{bmatrix} \mathbf{c}^T\mathbf{v}_1 \\ \mathbf{c}^T\mathbf{v}_2 \\ \vdots \\ \mathbf{c}^T\mathbf{v}_{12} \end{bmatrix} = \begin{bmatrix} 0 \\ 60 \\ 54 \\ 28 \\ 54 \\ 114 \\ 108 \\ 82 \\ 16 \\ 72 \\ 70 \\ 44 \end{bmatrix}$$

$$A_H \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \dots \mathbf{v}_{12} \end{bmatrix} = \begin{bmatrix} 0 & 18 & \dots & 20 \\ 0 & 6 & \dots & 6 \end{bmatrix}.$$

The linear program (11.10) would then be equivalent to

$$\max \quad 0\alpha_1 + 60\alpha_2 + \cdots + 44\alpha_{12}$$

s.t.

$$\begin{bmatrix} 0 & 18 & \dots & 20 \\ 0 & 6 & \dots & 6 \\ 1 & 1 & \dots & 1 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_{12} \end{bmatrix} = \begin{bmatrix} 20 \\ 10 \\ 1 \end{bmatrix} \tag{11.14}$$

$$\alpha_1, \alpha_2, \dots, \alpha_{12} \geq 0.$$

If $\boldsymbol{\alpha}$ is the optimal solution to (11.14), then

$$\mathbf{x} = \sum_{j=1}^{12} \alpha_j \mathbf{v}_j$$

is the optimal solution to (11.10).