16

LAGRANGIAN RELAXATION AND NETWORK OPTIMIZATION

I never missed the opportunity to remove obstacles in the way of unity.
—Mohandas Gandhi

Chapter Outline

16.1 INTRODUCTION

As we have noted throughout our discussion in this book, the basic network flow models that we have been studying—shortest paths, maximum flows, minimum cost flows, minimum spanning trees, matchings, and generalized and convex flows—arise in numerous applications. These core network models are also building blocks for many other models and applications, in the sense that many models met in practice have embedded network structure: that is, the broader models are network problems with additional variables and/or constraints.

In this chapter we consider ways to solve these models using a solution strategy known as *decomposition* which permits us to draw upon the many algorithms that we have developed in previous chapters to exploit the underlying network structure. In a sense this chapter serves a dual purpose. First, it permits us to introduce a broader set of network optimization models than we have been considering in our earlier discussion. As such, this chapter provides a glimpse of how network flow models arise in a wide range of applied problem settings that cannot be modeled as pure network flow problems. Second, the chapter introduces a solution method, known as *Lagrangian relaxation*, that has become one of the very few solution methods in optimization that cuts across the domains of linear and integer programming, combinatorial optimization, and nonlinear programming.

Perhaps the best way to understand the basic idea of Lagrangian relaxation is via an example.

Constrained Shortest Paths

Consider the network shown in Figure 16.1(a) which has two attributes associated with each arc (i, j): a cost c_{ij} and a traversal time t_{ij}. Suppose that we wish to find the shortest path from the source node 1 to the sink node 6, but we wish to restrict our choice of paths to those that require no more than $T = 10$ time units to traverse. This type of constrained shortest path application arises frequently in practice since in many contexts a company (e.g., a package delivery firm) wants to provide its services at the lowest possible cost and yet ensure a certain level of service to its customers (as embodied in the time restriction). In general, the constrained shortest path problem from node 1 to node n can be stated as the following integer programming problem:

$$\text{Minimize} \quad \sum_{(i,j)\in A} c_{ij}x_{ij} \tag{16.1a}$$

subject to

$$\sum_{\{j:(i,j)\in A\}} x_{ij} - \sum_{\{j:(j,i)\in A\}} x_{ji} = \begin{cases} 1 & \text{for } i = 1 \\ 0 & \text{for } i \in N - \{1, n\}, \\ -1 & \text{for } i = n \end{cases} \tag{16.1b}$$

$$\sum_{(i,j)\in A} t_{ij}x_{ij} \le T, \tag{16.1c}$$

$$x_{ij} = 0 \text{ or } 1 \qquad \text{for all } (i, j) \in A. \tag{16.1d}$$

The problem is not a shortest path problem because of the timing restriction. Rather, it is a shortest path problem with an additional side constraint (16.1c). Instead of solving this problem directly, suppose that we adopt an indirect approach by combining time and cost into a single *modified cost*; that is, we place a dollar equivalent on time. So instead of setting a limit on the total time we can take on the chosen path, we set a "toll charge" on each arc proportional to the time that it takes to

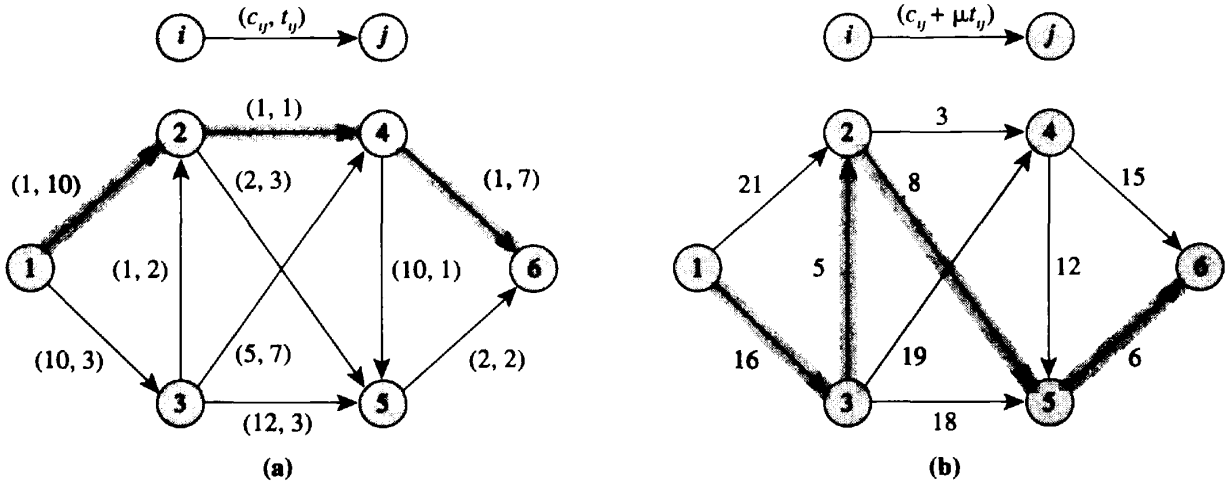

Figure 16.1 Time-constrained shortest path problem: (a) constrained shortest path problem (bold lines denote the shortest path for $\mu = 0$); (b) modified cost $c + \mu t$ with Lagrange multiplier $\mu = 2$ (bold lines denote the shortest path).

traverse that arc. For example, we might charge \$2 for each hour that it takes to traverse any arc. Note that if the toll charge is zero, we are ignoring time altogether and the problem becomes a usual shortest path problem with respect to the given costs. On the other hand, if the toll charge is very large, these charges become the dominant cost and we will be seeking the quickest path from the source to the sink. Can we find a toll charge somewhere in between these values so that by solving the shortest path problem with the combined costs (the toll charges and the original costs), we solve the constrained shortest path problem as a single shortest path problem?

For any choice μ of the toll charge, we solve a shortest path problem with respect to the modified costs $c_{ij} + \mu t_{ij}$. For the sample data shown in Figure 16.1(a), if $\mu = 0$, the modified problem becomes the shortest path problem with respect to the original costs c_{ij} and the shortest path 1–2–4–6 has length 3. This value is an obvious lower bound on the length of the constrained shortest path since it ignores the timing constraint. Now suppose that we set $\mu = 2$ and solve the modified problem. Figure 16.1(b) shows the modified costs $c_{ij} + 2t_{ij}$. The shortest path 1–3–2–5–6 has length 35. In this case, the path 1–3–2–5–6 that solves the modified problem happens to require 10 units to traverse, so it is a feasible constrained shortest path. Is it an optimal constrained shortest path?

To answer this question, let us make an important observation (which we will prove formally in the next section). Let P, with cost $c_P = \sum_{(i,j)\in P} c_{ij}$ and traversal time $t_P = \sum_{(i,j)\in P} t_{ij}$, be *any* feasible path to the constrained shortest path problem, and let $l(\mu)$ denote the optimal length of the shortest path with the modified costs when we impose a toll of μ units. Since the path P is feasible for the constrained shortest path problem, the time t_P required to traverse this path is at most $T = 10$ units. With respect to the modified costs $c_{ij} + \mu t_{ij}$, the cost $c_P + \mu t_P$ of the path P is the path's true cost c_P plus $\mu t_P \leq \mu T$ units. Therefore, if we subtract μT from the modified cost $c_P + \mu t_P$ of this path, we obtain a lower bound $c_P + \mu t_P - \mu T = c_P + \mu(t_P - T) \leq c_P$ on the cost c_P of this path. Since the shortest path with respect to the modified arc costs is less than or equal to the modified cost of any particular path, $l(\mu) \leq c_P + \mu t_P$ and so $l(\mu) - \mu T$ is a *common* lower bound on the length of *any* feasible path P and thus on the length of the constrained shortest path. Because this argument is completely general and applies to any value $\mu \geq 0$ of the toll charges, if we subtract μT from the optimal length of the shortest path of the modified problem, we obtain a lower bound on the optimal cost of the constrained shortest path problem.

Bounding Principle. *For any nonnegative value of the toll μ, the length $l(\mu)$ of the modified shortest path with costs $c_{ij} + \mu t_{ij}$ minus μT is a lower bound on the length of the constrained shortest path.*

Note that for our numerical example, for $\mu = 2$, the cost of the modified shortest path problem is 35 units and so $35 - 2(T) = 35 - 2(10) = 15$ is a lower bound on the length of the optimal constrained shortest path. But since the path 1–3–2–5–6 is a feasible solution to the constrained shortest path problem and its cost equals the lower bound of 15 units, we can be assured that it is an optimal constrained shortest path.

Observe that in this example we have been able to solve a difficult optimization

model (the constrained shortest path problem is an $\mathcal{NP}$-complete problem) by removing one or more problem constraints—in this case the single timing constraint—that makes the problem much more difficult to solve. Rather than solving the difficult optimization problem directly, we combined the complicating timing constraint with the original objective function, via the toll μ, so that we could then solve a resulting embedded shortest path problem. The motivation for adopting this approach was our observation that the original constrained shortest path problem had an attractive substructure, the shortest path problem, that we would like to exploit algorithmically. Whenever we can identify such attractive substructure, we could adopt a similar approach. For reasons that will become clearer in the next section, this general solution approach has become known as *Lagrangian relaxation*.

In our example we have been fortunate to find a constrained shortest path by solving the Lagrangian subproblem for a particular choice of the toll μ. We will not always be so lucky; nevertheless, as we will see, the lower bounding mechanism of Lagrangian relaxation frequently provides valuable information that we can exploit algorithmically.

Lagrangian relaxation is a general solution strategy for solving mathematical programs that permits us to decompose problems to exploit their special structure. As such, this solution approach is perfectly tailored for solving many models with embedded network structure. The Lagrangian solution strategy has a number of significant advantages:

1. Since it is often possible to decompose models in several ways and apply Lagrangian relaxation to each different decomposition, Lagrangian relaxation is a very flexible solution approach. Indeed, because of its flexibility, Lagrangian relaxation is more of a general problem solving strategy and solution framework than any single solution technique.
2. In decomposing problems, Lagrangian relaxation solves core subproblems as stand-alone models. Consequently, the solution approach permits us to exploit any known methodology or algorithm for solving the subproblems. In particular, when the subproblems are network models, the Lagrangian solution approach can take advantage of the various algorithms that we have developed previously in this book.
3. As we have already noted, Lagrangian relaxation permits us to develop bounds on the value of the optimal objective function and, frequently, to quickly generate good, though not necessarily optimal solutions with associated performance guarantees—that is, a bound on how far the solution could possibly be from optimality (in objective function value). In many instances in the context of integer programming, the bounds provided by Lagrangian relaxation methods are much better than those generated by solving the linear programming relaxation of the problems, and as a consequence, Lagrangian relaxation is often an attractive alternative to linear programming as a bounding mechanism in branch-and-bound methods for solving integer programs.
4. In many instances we can use Lagrangian relaxation methods to devise effective heuristic solution methods for solving complex combinatorial optimization problems and integer programs.

In the remainder of this chapter we describe the Lagrangian relaxation solution approach in more detail and demonstrate its use in solving several important network optimization models. Our purpose is not to present a comprehensive treatment of Lagrangian relaxation or of its applications to the field of network optimization, but rather to introduce this general solution strategy and to illustrate its applications in a way that would lay the essential foundations for applying the method in many other problem contexts. As a by-product of this discussion, in the text and in the exercises at the end of this chapter we introduce several noteworthy network optimization models that we do not treat elsewhere in the book.

Since one of the principal uses of Lagrangian relaxation is within implicit enumeration procedures for solving integer programs, before describing Lagrangian relaxation in more detail, we first discuss its use within classical branch-and-bound algorithms for solving integer programs. The reader can skip this section without loss of continuity.

16.2 PROBLEM RELAXATIONS AND BRANCH AND BOUND

In the last section we observed that Lagrangian relaxation permits us to develop a lower bound on the optimal length of a constrained shortest path. In Section 16.3 we develop a generalization of this result, showing that we can obtain a lower bound on the optimal objective function value of any minimization problem. These lower bounds can be of considerable value: for example, for our constrained shortest path example, we were able to use a lower bound to demonstrate that a particular solution that we generated by solving a shortest path subproblem, with modified costs, was optimal for the overall constrained problem. In general, we will not always be as fortunate in being able to use a lower bound to guarantee that the solution to a single subproblem solves the original problem. Nevertheless, as we show briefly in the section, we might still be able to use lower bounds as an algorithmic tool in reducing the number of computations required to solve combinatorial optimization problems formulated as integer programs.

Consider the following integer programming model:

Minimize cx

subject to

$$x \in F.$$

In this formulation, the set F represents the set of feasible solutions to an integer program, that is, the set of solutions $x = (x_1, x_2, \ldots, x_J)$ to the system

$$\mathcal{A}x = b,$$

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

In a certain conceptual sense, this integer program is trivial to solve: We simply enumerate every combination of the decision variables, that is, all zero–one vectors $(x_1, x_2, \ldots, x_J)$ obtained by setting each variable x_j to value zero or 1; from among

all those vectors that satisfy the given equality constraint $\mathcal{A}x = b$, we choose the combination with the smallest value of the objective function cx. Of course, because of its combinatorial explosiveness, this total enumeration procedure is limited to very small problems; for a problem with 100 decision variables, even if we could compute one solution every nanosecond (10^{-9} second), enumerating all 2^{100} solutions would take us over a million million years—that is, a million different million years!

Can we avoid any of these computations? Suppose that $F = F^1 \cup F^2$. For example, we might obtain F^1 from F by adding the constraint $x_1 = 0$ and F^2 by adding the constraint $x_1 = 1$. Note that the optimal solution over the feasible set F is the best of the optimal solutions over F^1 and F^2. Suppose that we already have found an optimal solution $\bar{x}$ to $\min\{cx : x \in F^2\}$ and that its objective function value is $z(\bar{x}) = 100$. The number of potential integer solutions in F^1 is still 2^{J-1}, so it will be prohibitively expensive to enumerate all these possibilities, except when J is small.

Rather than attempt to solve the integer program over the feasible region F^1, suppose that we solve a *relaxed* version of the problem, possibly by relaxing the integrality constraints, and/or possibly by performing a Lagrangian relaxation of the problem. In general, we obtain a relaxation by removing some constraints from the model: for example, by replacing the restrictions $x_j \geq 0$ and integer, by the restriction $x_j \geq 0$, or by deleting one or more constraints of the form $\alpha x = \beta$. We could use many different types of relaxation—in Lagrangian relaxation, for example, we not only delete some problem constraints, but we also change the objective function of the problem. For the purpose of this discussion, we merely require that we relax some of the problem constraints and that the objective function value of the relaxation is a lower bound on the objective function value of the original problem.

Let x' denote an optimal solution to the relaxation, and let $z(x')$ denote the objective function value of this solution. We consider four possibilities:

1. The solution x' does not exist because the relaxed problem has no feasible solution.
2. The solution x' happens to lie in F^1 (even though we relaxed some of the constraints).
3. The solution x' does not lie in F^1 and its objective function value $z(x')$ satisfies the inequality $z(x') \geq z(\bar{x}) = 100$.
4. The solution x' does not lie in F^1 and its objective function value $z(x')$ of x' satisfies the inequality $z(x') < z(\bar{x}) = 100$.

Note that these four alternatives exhaust all possible outcomes and are mutually exclusive. Therefore, exactly one of them must occur.

We now make an important observation. In cases 1 to 3, we can terminate our computations: we have solved the original problem over the set F, even though we have not explicitly solved any integer program (assuming that we obtained the solution over the set F^2 without solving an integer program). In case 1, since the relaxation of the set F^1 is empty, the set F^1 is also empty, so the solution $\bar{x}$ solves the original (overall) integer program. In case 2, since we have found the optimal solution in the relaxation (and so a superset) of the set F^1, and this solution lies in

F^1, we have also found the best solution in F^1; therefore, either $\bar{x}$ or x' is the solution to the original problem (whichever solution has the smaller objective function value). Note that in this case we have implicitly considered (enumerated) all of the solutions in F^1 in the sense that we know that no solution in this set is better than $\bar{x}$. In case 3, the solution $\bar{x}$ has as good an objective function value as the best solution in a relaxation of F^1, so it has an objective function value that is as good as any solution in F^1. Therefore, $\bar{x}$ solves the original problem. Note that in case 3 we have used *bounding* information on the objective function value to eliminate the solutions in the set F^1 from further consideration.

In case 4, we have not yet solved the original problem. We can either try to solve the problem minimize $\{cx : x \in F^1\}$ by some direct method of integer programming or, we can partition F^1 into two sets F^3 and F^4. For example, we might obtain F^3 from F by constraining $x_1 = 0$ and $x_2 = 0$ and obtain F^4 by setting $x_1 = 0$ and $x_2 = 1$. We could then apply any relaxation or direct approach for the problems defined over the sets F^3 and F^4.

In a general branch-and-bound procedure, we would systematically partition the feasible region F into subregions $F^1, F^2, F^3, \ldots, F^K$. Let $\bar{x}$ denote the best feasible solution (in objective function value) we have obtained in prior computations. Suppose that for each $k = 1, 2, \ldots, K$, either F^k is empty or x^k is a solution of a relaxation of the set F^k and $c\bar{x} \leq cx^k$. Then no point in any of the regions F^1, $F^2, F^3, \ldots, F^k$ could have a better objective function value than $\bar{x}$, so $\bar{x}$ solves the original optimization problem. If $c\bar{x} > cx^k$, though, for any region F^k, we would need to subdivide this region by "branching" on some of the variables (i.e., dividing a subregion in two by setting $x_j = 0$ or $x_j = 1$ for some variable j to define two new subregions). Whenever we have satisfied the test $c\bar{x} \leq cx^k$ for all of the subregions (or we know they are empty), we have solved the original problem.

The intent of the branch-and-bound method is to find an optimal solution by solving only a small number of relaxations. To do so, we would need to obtain good solutions quickly and obtain good relaxations so that the objective function value $z(x^k)$ of the solution x^k to the relaxation of the set F^k is close in objective function value to the optimal solution over F^k itself.

In practice, in implementing the branch-and-bound procedure, we need to make many design decisions concerning the order for choosing the subregions, the variables to branch on for each subregion, and mechanisms (e.g., heuristic procedures) that we might use to find "good" feasible solutions. The literature contains many clever approaches for resolving these issues and for designing branch-and-bound procedures that are quite effective in practice. We also need to develop good relaxations that would permit us to obtain effective (tight) lower bounds: if the lower bounds are weak, cases 2 and 3 will rarely occur and the branch-and-bound procedure will degenerate into complete enumeration. On the other hand, if the bounds are very tight, the relaxations will permit us to eliminate much of the enumeration and develop very effective solution procedures. Since our purpose in this chapter is to introduce one relaxation procedure that has proven to be very effective in practice and discuss its applications, we will not consider the detailed design choices for implementing the branch-and-bound procedure.

We next summarize the basic underlying ideas of the Lagrangian relaxation technique.

16.3 LAGRANGIAN RELAXATION TECHNIQUE

To describe the general form of the Lagrangian relaxation procedure, suppose that we consider the following generic optimization model formulated in terms of a vector x of decision variables:

$$z^* = \min cx$$

subject to

$$\mathcal{A}x = b, \qquad \text{(P)}$$
$$x \in X.$$

This model (P) has a linear objective function cx and a set $\mathcal{A}x = b$ of explicit linear constraints. The decision variables x are also constrained to lie in a given constraint set X which, as we will see, often models embedded network flow structure. For example, the constraint set $X = \{x : \mathcal{N}x = q, 0 \leq x \leq u\}$ might be all the feasible solutions to a network flow problem with a supply/demand vector q. Or, the set X might contain the incidence vectors of all spanning trees or matchings of a given graph. Unless we state otherwise, we assume that the set X is finite (e.g., for network flow problems, we will let it be the finite set of spanning tree solutions).

As its name suggests, the Lagrangian relaxation procedure uses the idea of relaxing the explicit linear constraints by bringing them into the objective function with associated Lagrange multipliers μ (this old idea might be a familiar one from advanced calculus in the context of solving nonlinear optimization problems). We refer to the resulting problem

$$\text{Minimize} \quad cx + \mu(\mathcal{A}x - b)$$

subject to

$$x \in X,$$

as a *Lagrangian relaxation* or *Lagrangian subproblem* of the original problem, and refer to the function

$$L(\mu) = \min\{cx + \mu(\mathcal{A}x - b) : x \in X\},$$

as the *Lagrangian function*. Note that since in forming the Lagrangian relaxation, we have eliminated the constraints $\mathcal{A}x = b$ from the problem formulation, the solution of the Lagrangian subproblem need not be feasible for the original problem (P). Can we obtain any useful information about the original problem even when the solution to the Lagrangian subproblem is not feasible in the original problem (P)? The following elementary observation is a key result that helps to answer this question and that motivates the use of the Lagrangian relaxation technique in general.

Lemma 16.1 (Lagrangian Bounding Principle). *For any vector μ of the Lagrangian multipliers, the value $L(\mu)$ of the Lagrangian function is a lower bound on the optimal objective function value z^* of the original optimization problem* (P).

Proof. Since $\mathcal{A}x = b$ for every feasible solution to (P), for any vector μ of Lagrangian multipliers, $z^* = \min\{cx : \mathcal{A}x = b, x \in X\} = \min\{cx + \mu(\mathcal{A}x - b) : \mathcal{A}x = b, x \in X\}$. Since removing the constraints $\mathcal{A}x = b$ from the second formulation

cannot lead to an increase in the value of the objective function (the value might decrease), $z^* \geq \min\{cx + \mu(\mathcal{A}x - b) : x \in X\} = L(\mu)$. ◆

As we have seen, for any value of the Lagrangian multiplier μ, $L(\mu)$ is a lower bound on the optimal objective function value of the original problem. To obtain the sharpest possible lower bound, we would need to solve the following optimization problem

$$L^* = \max_{\mu} L(\mu)$$

which we refer to as the *Lagrangian multiplier problem* associated with the original optimization problem (P). The Lagrangian bounding principle has the following immediate implication.

Property 16.2 (Weak Duality). *The optimal objective function value L^* of the Lagrangian multiplier problem is always a lower bound on the optimal objective function value of the problem* (P) *(i.e., $L^* \leq z^*$).*

Our preceding discussion provides us with valid bounds for comparing objective function values of the Lagrange multiplier problem and optimization (P) for any choices of the Lagrange multipliers μ and any feasible solution x of (P):

$$L(\mu) \leq L^* \leq z^* \leq cx.$$

These inequalities furnish us with a guarantee when a Lagrange multiplier μ to the Lagrange multiplier problem or a feasible solution x to the original problem (P) are optimal.

Property 16.3 (Optimality Test)

(a) Suppose that μ is a vector of Lagrangian multipliers and x is a feasible solution to the optimization problem (P) *satisfying the condition $L(\mu) = cx$. Then $L(\mu)$ is an optimal solution of the Lagrangian multiplier problem [i.e., $L^* = L(\mu)$] and x is an optimal solution to the optimization problem* (P).

(b) If for some choice of the Lagrangian multiplier vector μ, the solution x^* of the Lagrangian relaxation is feasible in the optimization problem (P), *then x^* is an optimal solution to the optimization problem* (P) *and μ is an optimal solution to the Lagrangian multiplier problem.*

Note that by assumption in part (b) of this property, $L(\mu) = cx^* + \mu(\mathcal{A}x^* - b)$ and $\mathcal{A}x^* = b$. Therefore, $L(\mu) = cx^*$ and part (a) implies that x^* solves problem (P) and μ solves the Lagrangian multiplier problem.

As indicated by Property 16.3, the bounding principle immediately implies one advantage of the Lagrangian relaxation approach—the method can give us a *certificate* [in the form of the equality $L(\mu) = cx$ for some Lagrange multiplier μ] for guaranteeing that a given feasible solution x to the optimization problem (P) is an optimal solution. Even if $L(\mu) < cx$, having the lower bound permits us to state a bound on how far a given solution is from optimality: If $[cx - L(\mu)]/L(\mu) \leq 0.05$, for example, we know that the objective function value of the feasible solution x is no more than 5% from optimality. This type of bound is very useful in practice—it

permits us to assess the degree of suboptimality of given solutions and it permits us to terminate our search for an optimal solution when we have a solution that we know is close enough to optimality (in objective function value) for our purposes.

Lagrangian Relaxation and Inequality Constraints

In the optimization model (P), the constraints $\mathcal{A}x = b$ are all equality constraints. In practice, we often encounter models, such as the constrained shortest path problem, that are formulated more naturally in inequality form $\mathcal{A}x \leq b$. The Lagrangian multiplier problem for these problems is a slight variant of the one we have just introduced: The Lagrangian multiplier problem becomes

$$L^* = \max_{\mu \geq 0} L(\mu).$$

That is, the only change in the Lagrangian multiplier problem is that the Lagrangian multipliers now are restricted to be nonnegative. In Exercise 16.1, by introducing "slack variables" to formulate the inequality problem as an equivalent equality problem, we show how to obtain this optimal multiplier problem from the one we have considered for the equality problem. This development implies that the bounding property, the weak duality property, and the optimality test 16.3(a) are valid when we apply Lagrangian relaxation to any combination of equality and inequality constraints.

There is, however, one substantial difference between relaxing equality constraints and inequality constraints. When we relax inequality constraints $\mathcal{A}x \leq b$, if the solution x^* of the Lagrangian subproblem happens to satisfy these constraints, it need *not* be optimal (see Exercise 16.2). In addition to being feasible, this solution needs to satisfy the *complementary slackness condition* $\mu(\mathcal{A}x^* - b) = 0$, which is familiar to us from much of our previous discussion of network flows in section 9.4.

Property 16.4. *Suppose that we apply Lagrangian relaxation to the optimization problem* ($P^{\leq}$) *defined as minimize* $\{cx : \mathcal{A}x \leq b \text{ and } x \in X\}$ *by relaxing the inequalities* $\mathcal{A}x \leq b$. *Suppose, further, that for some choice of the Lagrangian multiplier vector* μ, *the solution* x^* *of the Lagrangian relaxation* (1) *is feasible in the optimization problem* ($P^{\leq}$), *and* (2) *satisfies the complementary slackness condition* $\mu(\mathcal{A}x^* - b) = 0$. *Then* x^* *is an optimal solution to the optimization problem* ($P^{\leq}$).

Proof. By assumption, $L(\mu) = cx^* + \mu(\mathcal{A}x^* - b)$. Since $\mu(\mathcal{A}x^* - b) = 0$, $L(\mu) = cx^*$. Moreover, since $\mathcal{A}x^* \leq b$, x^* is feasible, and so by Property 16.3(a) x^* solves problem ($P^{\leq}$). ◆

Are solutions to the Lagrangian subproblem of use in solving the original problem? Properties 16.3 and 16.4 show that certain solutions of the Lagrangian subproblem provably solve the original problem. We might distinguish two other cases: (1) when solutions obtained by relaxing inequality constraints are feasible but are not provably optimal for the original problem (since they do not satisfy the complementary slackness condition), and (2) when solutions to the Lagrangian relaxation are not feasible in the original problem.

In the first case, the solutions are candidate optimal solutions (possibly for use

in a branch-and-bound procedure). In the second case, for many applications, researchers have been able to devise methods to modify "modestly" infeasible solutions so that they become feasible with only a slightly degradation in the objective function value. These observations suggest that we might be able to use the solutions obtained from the Lagrangian subproblem as "approximate" solutions to the original problem, even when they are not provably optimal; in these instances, we can use Lagrangian relaxation as a heuristic method for generating provably good solutions in practice (the solutions might be provably good because of the Lagrangian lower bound information). The development of these heuristic methods depends heavily on the problem context we are studying, so we will not attempt to provide any further details.

Solving the Lagrangian Multiplier Problem

How might we solve the Lagrangian multiplier problem? To develop an understanding of possible solution techniques, let us consider the constrained shortest path problem that we defined in Section 16.1. Suppose that now we have a time limitation of $T = 14$ instead of $T = 10$. When we relax the time constraint, the Lagrangian multiplier function $L(\mu)$ for the constrained shortest path problem becomes

$$L(\mu) = \min\{c_P + \mu(t_P - T) : P \in \mathcal{P}\}.$$

In this formulation, $\mathcal{P}$ is the collection of all directed paths from the source node 1 to the sink node n. For convenience, we refer to the quantity $c_P + \mu(t_P - T)$ as the *composite cost* of the path P. For a specific value of the Lagrangian multiplier μ, we can solve $L(\mu)$ by enumerating all the directed paths in $\mathcal{P}$ and choosing the path with the smallest composite cost. Consequently, we can solve the Lagrangian multiplier problem by determining $L(\mu)$ for all nonnegative values of the Lagrangian multiplier μ and choosing the value that achieves $\max_{\mu \geq 0} L(\mu)$.

Let us illustrate this brute force approach geometrically. Figure 16.2 records the cost and time data for every path for our numerical example. Note that the composite cost $c_P + \mu(t_P - T)$ for any path P is a linear function of μ with an intercept of c_P and a slope of $(t_P - T)$. In Figure 16.3 we have plotted each of these path composite cost functions. Note that for any specific value of the Lagrange multiplier μ, we can find $L(\mu)$ by evaluating each composite cost function (line) and identifying the one with the least cost. This observation implies that the Lagrangian multiplier function $L(\mu)$ is the lower envelope of the composite cost lines and that the highest point on this envelope corresponds to the optimal solution of the Lagrangian multiplier problem.

In practice, we would never attempt to solve the problem in this way because the number of directed paths from the source node to the sink node typically grows exponentially in the number of nodes in the underlying network, so any such enumeration procedure would be prohibitively expensive. Nevertheless, this problem geometry helps us to understand the nature of the Lagrangian multiplier problem and suggests methods for solving the problem.

As we noted in the preceding paragraph, to find the optimal multiplier value μ^* of the Lagrangian multiplier problem, we need to find the highest point of the Lagrangian multiplier function $L(\mu)$. Suppose that we consider the polyhedron de-

Path P	Path cost c_P	Path time t_P	Composite cost $c_P + \mu (t_P - T)$
1–2–4–6	3	18	$3 + 4\mu$
1–2–5–6	5	15	$5 + \mu$
1–2–4–5–6	14	14	14
1–3–2–4–6	13	13	$13 - \mu$
1–3–2–5–6	15	10	$15 - 4\mu$
1–3–2–4–5–6	24	9	$24 - 5\mu$
1–3–4–6	16	17	$16 + 3\mu$
1–3–4–5–6	27	13	$27 - \mu$
1–3–5–6	24	8	$24 - 6\mu$

Figure 16.2 Path cost and time data for constrained shortest path example with $T = 14$.

fined by those points that lie on or below the function $L(\mu)$. These are the shaded points in Figure 16.3. Then geometrically, we are finding the highest point in a polyhedron defined by the function $L(\mu)$, which is a linear program.

Even though we have illustrated this property on a specific example, this situation is completely general. Consider the generic optimization model (P), defined

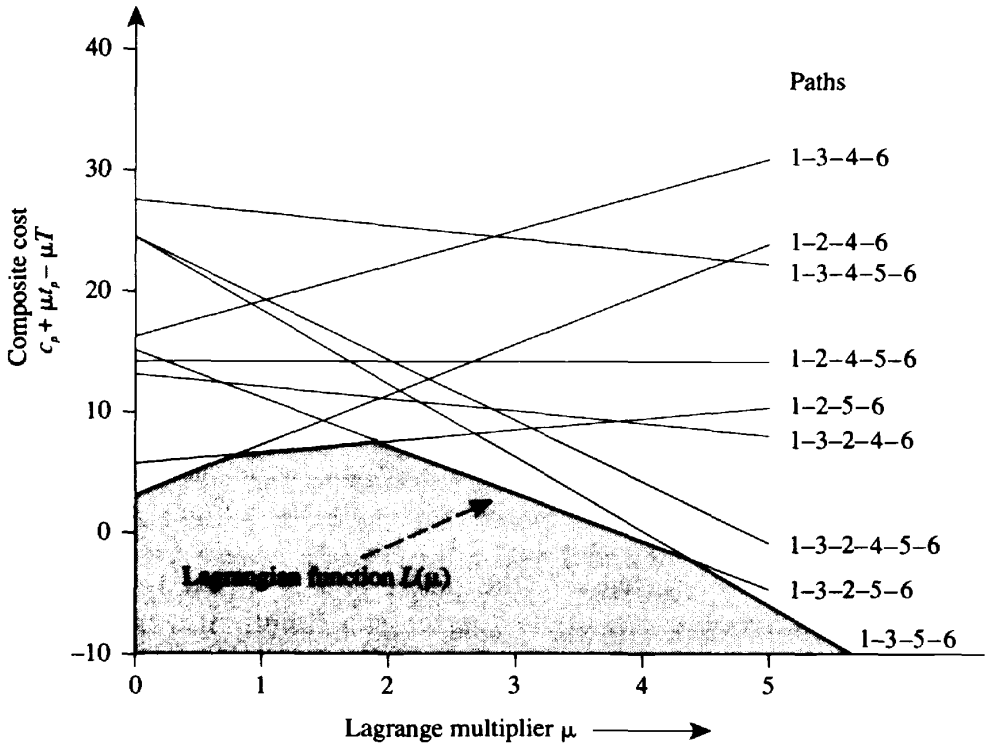

Figure 16.3 Lagrangian function for $T = 14$.

as $\min\{cx : \mathcal{A}x = b, x \in X\}$ and suppose that the set $X = \{x^1, x^2, \ldots, x^K\}$ is finite. By relaxing the constraints $\mathcal{A}x = b$, we obtain the Lagrangian multiplier function $L(\mu) = \min\{cx + \mu(\mathcal{A}x - b) : x \in X\}$. By definition,

$$L(\mu) \leq cx^k + \mu(\mathcal{A}x^k - b) \qquad \text{for all } k = 1, 2, \ldots, K.$$

In the space of composite costs and Lagrange multipliers μ (as in Figure 16.3), each function $cx^k + \mu(\mathcal{A}x^k - b)$ is a multidimensional "line" called a *hyperplane* (if μ is two-dimensional, it is a plane). The Lagrangian multiplier function $L(\mu)$ is the lower envelope of the hyperplanes $cx^k + \mu(\mathcal{A}x^k - b)$ for $k = 1, 2, \ldots, K$. In the Lagrangian multiplier problem, we wish to determine the highest point on this envelope: We can find this point by solving the optimization problem

$$\text{Maximize} \quad w$$

subject to

$$w \leq cx^k + \mu(\mathcal{A}x^k - b) \qquad \text{for all } k = 1, 2, \ldots, K,$$

$$\mu \text{ unrestricted},$$

which is clearly a linear program. We state this result as a theorem.

Theorem 16.5. *The Lagrangian multiplier problem* $L^* = \max_\mu L(\mu)$ *with* $L(\mu) = \min\{cx^k + \mu(\mathcal{A}x - b) : x \in X\}$ *is equivalent to the linear programming problem* $L^* = \max\{w : w \leq cx^k + \mu(\mathcal{A}x^k - b)$ *for* $k = 1, 2, \ldots, K\}$. ◆

Since, as shown by the preceding theorem, the Lagrangian multiplier problem is a linear program, we could solve this problem by applying the linear programming methodology. One resulting algorithm, which is known as *Dantzig–Wolfe decomposition* or *generalized linear programming*, is an important solution methodology that we discuss in some depth in Chapter 17 in the context of solving the multicommodity flow problem. One of the disadvantages of this approach is that it requires the solution of a series of linear programs that are rather expensive computationally. Another approach might be to apply some type of gradient method to the Lagrangian function $L(\mu)$. As shown by the constrained shortest path example, the added complication of this approach is that the Lagrangian function $L(\mu)$ is not differentiable. It is differentiable whenever the optimal solution of the Lagrangian subproblem is unique; but when the subproblem has two or more solutions, the Lagrangian function generally is not differentiable. For example, in Figure 16.4, at $\mu = 0$, the path 1–2–4–6 is the unique shortest path solution to the subproblem and the function $L(\mu)$ is differentiable. At this point, for the path P = 1–2–4–6, $L(\mu) = c_P + \mu(t_P - T)$; since $t_P = 18$ and $T = 14$, $L(\mu)$ has a slope $(t_P - T) = (18 - 14) = 4$. At the point $\mu = 2$, however, the paths 1–2–5–6 and 1–3–2–5–6 both solve the Lagrangian subproblem and the Lagrangian function is not differentiable. To accommodate these situations, we next describe a technique, known as the *subgradient optimization technique*, for solving the (nondifferentiable) Lagrangian multiplier problem.

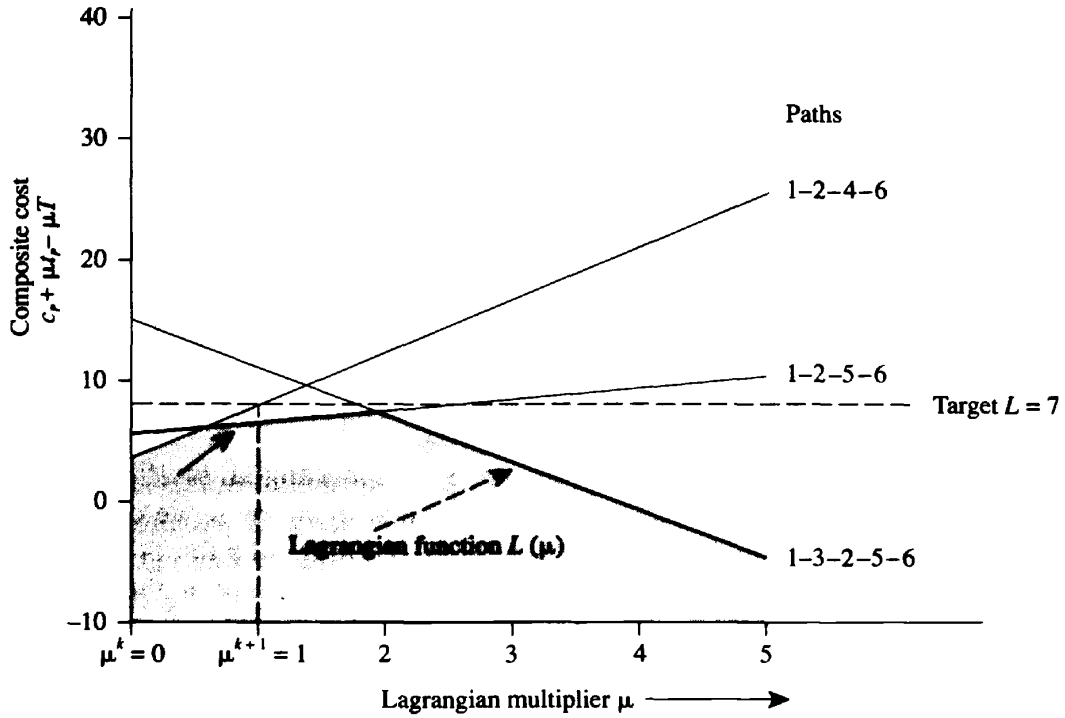

Figure 16.4 Steps of Newton's method for $T = 14$.

Subgradient Optimization Technique

In solving optimization problems with the nonlinear objective function $f(x)$ of an n-dimensional vector x, researchers and practitioners often use variations of the following classical idea: Form the gradient $\nabla f(x)$ of f defined as a row vector with components $(\partial f(x)/dx_1, \partial f(x)/dx_2, \ldots, \partial f(x)/dx_n)$. Recall from advanced calculus that the directional derivative of f in the direction d satisfies the equality

$$\lim_{\theta \to 0} \frac{f(x + \theta d) - f(x)}{\theta} = \nabla f(x)d.$$

So if we choose the direction d so that $\nabla f(x)d > 0$ and move in the direction d with a small enough "step length" θ—that is, change x to $x + \theta d$—we move uphill. This simple observation lies at the core of a considerable literature in nonlinear programming known as *gradient methods*.

Suppose that in solving the Lagrangian multiplier problem, we are at a point where the Lagrangian function $L(\mu) = \min\{cx + \mu(\mathcal{A}x - b) : x \in X\}$ has a unique solution $\bar{x}$, so is differentiable. Since $L(\mu) = c\bar{x} + \mu(\mathcal{A}\bar{x} - b)$ and the solution $\bar{x}$ remains optimal for small changes in the value of μ, the gradient at this point is $\mathcal{A}\bar{x} - b$, so a gradient method would change the value of μ as follows:

$$\mu \leftarrow \mu + \theta(\mathcal{A}\bar{x} - b).$$

In this expression, θ is a step size (a scalar) that specifies how far we move in the gradient direction. Note that this procedure has a nice intuitive interpretation. If $(\mathcal{A}\bar{x} - b)_i = 0$, the solution x uses up exactly the required units of the ith resource, and we hold the Lagrange multiplier (the toll) μ_i of that resource at its current value;

if $(\mathcal{A}\bar{x} - b)_i < 0$, the solution x uses up less than the available units of the ith resource and we decrease the Lagrange multiplier μ_i on that resource; and if $(\mathcal{A}\bar{x} - b)_i > 0$, the solution x uses up more than the available units of the ith resource and we increase the Lagrange multiplier μ_i on that resource.

To solve the Lagrangian multiplier problem, we adopt a rather natural extension of this solution approach. We let μ^0 be any initial choice of the Lagrange multiplier; we determine the subsequent values μ^k for $k = 1, 2, \ldots$, of the Lagrange multipliers as follows:

$$\mu^{k+1} = \mu^k + \theta_k(\mathcal{A}x^k - b).$$

In this expression, x^k is any solution to the Lagrangian subproblem when $\mu = \mu^k$ and θ_k is the step length at the kth iteration.

To ensure that this method solves the Lagrangian multiplier problem, we need to exercise some care in the choice of the step sizes. If we choose them too small, the algorithm would become stuck at the current point and not converge; if we choose the step sizes too large, the iterates μ^k might overshoot the optimal solution and perhaps even oscillate between two nonoptimal solutions (see Exercise 16.4 for an example). The following compromise ensures that the algorithm strikes an appropriate balance between these extremes and does converge:

$$\theta_k \to 0 \quad \text{and} \quad \sum_{j=1}^{k} \theta_j \to \infty.$$

For example, choosing $\theta_k = 1/k$ satisfies these conditions. These conditions ensure that the algorithm always converges to an optimal solution of the multiplier problem, but a proof of this convergence result is beyond the scope of our coverage in this book (the reference notes cite papers and books that examine the convergence of subgradient methods).

One important variant of the subgradient optimization procedure would be an adaptation of "Newton's method" for solving systems of nonlinear equations. Suppose, as before, that $L(\mu^k) = cx^k + \mu^k(\mathcal{A}x^k - b)$; that is, x^k solves the Lagrangian subproblem when $\mu = \mu^k$. Suppose that we assume that x^k continues to solve the Lagrangian subproblem as we vary μ; or, stated in another way, we make a linear approximation $r(\mu) = cx^k + \mu(\mathcal{A}x^k - b)$ to $L(\mu)$. Suppose further that we know the optimal value L^* of the Lagrangian multiplier problem (which we do not). Then we might move in the subgradient direction until the value of the linear approximation exactly equals L^*. Figure 16.4 shows an example of this procedure when applied to our constrained shortest path example, starting with $\mu^k = 0$. At this point, the path P = 1–2–4–6 solves the Lagrangian subproblem and $\mathcal{A}x^k - b$ equals $t_P - T = 18 - 14 = 4$. Since $L^* = 7$ and the path P has a cost $c_P = 3$, in accordance with this linear approximation, or Newton's method, we would approximate $L(\mu)$ by $r(\mu) = 3 + 4\mu$, set $3 + 4\mu = 7$, and define the new value of μ as $\mu^{k+1} = (7 - 3)/4 = 1$. In general, we set the step length θ_k so that

$$r(\mu^{k+1}) = cx^k + \mu^{k+1}(\mathcal{A}x^k - b) = L^*,$$

or since, $\mu^{k+1} = \mu^k + \theta_k(\mathcal{A}x^k - b)$,

$$r(\mu^{k+1}) = cx^k + [\mu^k + \theta_k(\mathcal{A}x^k - b)](\mathcal{A}x^k - b) = L^*.$$

Collecting terms, recalling that $L(\mu^k) = cx^k + \mu(\mathcal{A}x^k - b)$, and letting $\| y \| = (\sum_j y_j^2)^{1/2}$ denote the Euclidean norm of the vector y, we can solve for the step length and find that

$$\theta_k = \frac{L^* - L(\mu^k)}{\| \mathcal{A}x^k - b \|^2}.$$

Since we do not know the optimal objective function value L^* of the Lagrangian multiplier problem (after all, that's what we are trying to find), practitioners of Lagrangian relaxation often use the following popular heuristic for selecting the step length:

$$\theta_k = \frac{\lambda_k[\text{UB} - L(\mu^k)]}{\| \mathcal{A}x^k - b \|^2}.$$

In this expression, UB is an upper bound on the optimal objective function value z^* of the problem (P), and so an upper bound on L^* as well, and λ_k is a scalar chosen (strictly) between 0 and 2. Initially, the upper bound is the objective function value of any known feasible solution to the problem (P). As the algorithm proceeds, if it generates a better (i.e., lower cost) feasible solution, it uses the objective function value of this solution in place of the upper bound UB. Usually, practitioners choose the scalars λ_k by starting with $\lambda_k = 2$ and then reducing λ_k by a factor of 2 whenever the best Lagrangian objective function value found so far has failed to increase in a specified number of iterations. Since this version of the algorithm has no convenient stopping criteria, practitioners usually terminate it after it has performed a specified number of iterations.

The rationale for these choices of the step size and the convergence proof of the subgradient method would take us beyond the scope of our coverage. In passing, we might note that the subgradient optimization procedure is not the only way to solve the Lagrangian multiplier problem: practitioners have used a number of other heuristics, including methods known as *multiplier ascent methods* that are tailored for special problems. Since we merely wish to introduce some of the basic concepts of Lagrangian relaxation and to indicate some of the essential methods used to solve the Lagrangian multiplier problem, we will not discuss these alternative methods.

Subgradient Optimization and Inequality Constraints

As we noted earlier in this section, if we apply Lagrangian relaxation to a problem with constraints $\mathcal{A}x \leq b$ stated in inequality form instead of the equality constraints, the Lagrange multipliers μ are constrained to be nonnegative. The update formula $\mu^{k+1} = \mu^k + \theta_k(\mathcal{A}x^k - b)$ might cause one or more of the components μ_i of μ to become negative. To avoid this possibility, we modify the update formula as follows:

$$\mu^{k+1} = [\mu^k + \theta_k(\mathcal{A}x^k - b)]^+.$$

In this expression, the notation $[y]^+$ denotes the "positive part" of the vector y; that is, the ith component of $[y]^+$ equals the maximum of 0 and y_i. Stated in another way, if the update formula $\mu^{k+1} = \mu^k + \theta_k(\mathcal{A}x^k - b)$ would cause the ith component of μ_i to be negative, then we simply set the value of this component to be zero. We then implement all the other steps of the subgradient procedure (i.e., the choice of

the step size θ at each step and the solution of the Lagrangian subproblems) exactly the same as for problems with equality constraints. For problems with both equality and inequality constraints, we use a straightforward mixture of the equality and inequality versions of the algorithm: whenever the update formula for the Lagrange multipliers would cause any component μ_i of μ corresponding to an inequality constraint to become negative, we set the value of that multiplier to be zero.

Let us illustrate the subgradient method for inequality constraints on our constrained shortest path example. Suppose that we start to solve our constrained shortest path problem at $\mu^0 = 0$ with $\lambda^0 = 0.8$ and with UB = 24, the cost corresponding to the shortest path 1–3–5–6 joining nodes 1 and 6. Suppose that we choose to reduce the scalar λ_k by a factor of 2 whenever three successive iterations at a given value of λ_k have not improved on the best Lagrangian objective function value $L(\mu)$. As we have already noted, the solution x^0 to the Lagrangian subproblem with $\mu = 0$ corresponds to the path P = 1–2–4–6, the Lagrangian subproblem has an objective function value of $L(0) = 3$, and the subgradient $\mathcal{A}x^0 - b$ at $\mu = 0$ is $(t_P - 14) = 18 - 14 = 4$. So at the first step, we choose

$$\theta_0 = 0.8(24 - 3)/16 = 1.05,$$

$$\mu^1 = [0 + 1.05(4)]^+ = 4.2.$$

For this value of the Lagrange multiplier, from Figure 16.3, we see that the path P = 1–3–2–5–6 solves the Lagrangian subproblem; therefore, $L(4.2) = 15 + 4.2(10) - 4.2(14) = 15 - 16.8 = -1.8$, and $\mathcal{A}x^1 - b$ equals $(t_P - 14) = 10 - 14 = -4$. Since the path 1–3–2–5–6 is feasible, and its cost of 15 is less than UB, we change UB to value 15. Therefore,

$$\theta_1 = 0.8(15 + 1.8)/16 = 0.84,$$

$$\mu^2 = [4.2 + 0.84(-4)]^+ = 0.84.$$

From iterations 2 through 5, the shortest paths alternate between the paths 1–2–4–6 and 1–3–2–5–6. At the end of the fifth iteration, the algorithm has not improved upon (increased) the best Lagrangian objective function value of 6.36 for three iterations, so we reduce λ_k by a factor of 2. In the next 7 iterations the shortest paths are the paths 1–2–5–6, 1–3–5–6, 1–3–2–5–6, 1–3–2–5–6, 1–2–5–6, 1–3–5–6, and 1–3–2–5–6. Once again for three consecutive iterations, the algorithm has not improved the best Lagrangian objective function value, so we decrease λ_k by a factor of 2 to value 0.2. From this point on, the algorithm chooses either path 1–3–2–5–6 or path 1–2–5–6 as the shortest path at each step. Figure 16.5 shows the first 33 iterations of the subgradient algorithm. As we see, the Lagrangian objective function value is converging to the optimal value $L^* = 7$ and the Lagrange multiplier is converging to its optimal value of $\mu^* = 2$.

Note that for this example, the optimal multiplier objective function value of $L^* = 7$ is strictly less than the length of the shortest constrained path, which has value 13. In these instances, we say that the Lagrangian relaxation has a *duality (relaxation) gap*. To solve problems with a duality gap to completion (i.e., to find an optimal solution and a guarantee that it is optimal), we would apply some form of enumeration procedure, such as branch and bound, using the Lagrangian lower bound to help reduce the amount of concentration required.

k	μ^k	$t_p - T$	$L(\mu^k)$	λ_k	θ_k
0	0.0000	4	3.0000	0.80000	1.0500
1	4.2000	−4	−1.8000	0.80000	0.8400
2	0.8400	4	6.3600	0.80000	0.4320
3	2.5680	−4	4.7280	0.80000	0.5136
4	0.5136	4	5.0544	0.80000	0.4973
5	2.5027	−4	4.9891	0.40000	0.2503
6	1.5016	1	6.5016	0.40000	3.3993
7	4.9010	−6	−5.4059	0.40000	0.2267
8	3.5406	−4	0.8376	0.40000	0.3541
9	2.1244	−4	6.5026	0.40000	0.2124
10	1.2746	1	6.2746	0.40000	3.4902
11	4.7648	−6	−4.5886	0.40000	0.2177
12	3.4589	−4	1.1646	0.20000	0.1729
13	2.7671	−4	3.9316	0.20000	0.1384
14	2.2137	−4	6.1453	0.20000	0.1107
15	1.7709	1	6.7709	0.20000	1.6458
16	3.4167	−4	1.3330	0.20000	0.1708
17	2.7334	−4	4.0664	0.20000	0.1367
18	2.1867	−4	6.2531	0.10000	0.0547
19	1.9680	1	6.9680	0.10000	0.8032
20	2.7712	−4	3.9150	0.10000	0.0693
21	2.4941	−4	5.0235	0.10000	0.0624
22	2.2447	−4	6.0212	0.05000	0.0281
23	2.1325	−4	6.4701	0.05000	0.0267
24	2.0258	−4	6.8966	0.05000	0.0253
25	1.9246	1	6.9246	0.00250	0.0202
26	1.9447	1	6.9447	0.00250	0.0201
27	1.9649	1	6.9649	0.00250	0.0201
28	1.9850	1	6.9850	0.00250	0.0200
29	2.0050	−4	6.9800	0.00250	0.0013
30	2.0000	−4	7.0000	0.00250	0.0012
31	1.9950	1	6.9950	0.00250	0.0200
32	2.0150	−4	6.9400	0.00250	0.0013
33	2.0100	−4	6.9601	0.00125	0.0006

Figure 16.5 Subgradient optimization for a constrained shortest path problem.

16.4 *LAGRANGIAN RELAXATION AND LINEAR PROGRAMMING*

In this section we discuss several theoretical properties of the Lagrangian relaxation technique. As we have noted earlier in Section 16.2, the primary use of the Lagrangian relaxation technique is to obtain lower bounds on the objective function values of (discrete) optimization problems. By relaxing the integrality constraints in the integer programming formulation of a discrete optimization problem, thereby

creating a linear programming relaxation, we obtain an alternative method for generating a lower bound. Which of these lower bounds is sharper (i.e., larger in value)? In this section we answer this question by showing that the lower bound obtained by the Lagrangian relaxation technique is at least as sharp as that obtained by using a linear programming relaxation. As a result, and because the Lagrangian relaxation bound is often easier to obtain than the linear programming relaxation bound, Lagrangian relaxation has become a very useful lower bounding technique in practice.

The content in this section requires some background in linear algebra and linear programming. We refer the reader to Appendix C for a review of this material.

Our first result in this section concerns the application of Lagrangian relaxation to a linear programming problem.

Theorem 16.6. *Suppose that we apply the Lagrangian relaxation technique to a linear programming problem* (P′) *defined as* $\min\{cx : \mathcal{A}x = b, \mathcal{D}x \leq q, x \geq 0\}$ *by relaxing the constraints* $\mathcal{A}x = b$. *Then the optimal value* L^* *of the Lagrangian multiplier problem equals the optimal objective function value of* (P′).

Proof. We use linear programming optimality conditions to prove the theorem. Suppose that x^* is an optimal solution of the linear programming problem (P′) and that π^* and γ^* denote vectors of optimal dual variables associated with the constraints $\mathcal{A}x = b$ and $\mathcal{D}x \leq q$. By linear programming theory, x^*, π^*, and γ^* satisfy the following dual feasibility and complementary slackness conditions:

$$c + \pi^*\mathcal{A} + \gamma^*\mathcal{D} \geq 0, \quad [c + \pi^*\mathcal{A} + \gamma^*\mathcal{D}]x^* = 0, \quad \text{and} \quad \gamma^*[\mathcal{D}x - q] = 0.$$

Consider the Lagrangian subproblem $L(\mu)$ at $\mu = \pi^*$, which is $L(\pi^*) = \min\{cx + \pi^*(\mathcal{A}x - b) : \mathcal{D}x \leq q, x \geq 0\}$. Notice that x^* is feasible for this problem because it is feasible to (P′). Moreover, for the fixed value $\mu = \pi^*$, the previous dual feasibility and complementary slackness conditions are exactly those for the Lagrangian subproblem; therefore, x^* also solves the Lagrangian subproblem at $\mu = \pi^*$. But since $\pi^*(\mathcal{A}x^* - b) = 0$, $L(\pi^*) = cx^*$. Consequently, Property 16.3 implies that $L^* = L(\pi^*) = cx^*$, the optimal objective function value of (P′). ◆

The preceding theorem shows that the Lagrangian relaxation technique provides an alternative method for solving a linear programming problem. Instead of solving the linear programming problem directly using any linear programming algorithm, we can relax a subset of the constraints and solve the Lagrangian multiplier problem by using subgradient optimization and solving a sequence of relaxed problems. In some situations the relaxed problem is easy to solve, but the original problem is not; in these situations, a Lagrangian relaxation-based algorithm is an attractive solution approach.

Suppose next that we apply Lagrangian relaxation to a discrete optimization problem (P) defined as $\min\{cx : \mathcal{A}x = b, x \in X\}$. We assume that the discrete set X is specified as $X = \{x : \mathcal{D}x \leq q, x \geq 0 \text{ and integer}\}$ for an integer matrix $\mathcal{D}$ and an integer vector q. Consequently, the problem (P) becomes

$$z^* = \min\{cx : \mathcal{A}x = b, \mathcal{D}x \leq q, x \geq 0 \text{ and integer}\}. \qquad \text{(P)}$$

We incur essentially no loss of generality by specifying the set X in this manner because we can formulate almost all real-life discrete optimization problems as integer programming problems. Let (LP) denote the linear programming relaxation of the problem (P) and let z° denote its optimal objective function value. That is,

$$z^\circ = \min\{cx : \mathcal{A}x = b, \mathcal{D}x \leq q, x \geq 0\}. \qquad \text{(LP)}$$

Clearly, $z^\circ \leq z^*$ because the set of feasible solutions of (P) lies within the set of feasible solutions of (LP). Therefore, the linear programming relaxation provides a valid lower bound on the optimal objective function value of (P). We have earlier shown in Property 16.2 that the Lagrangian multiplier problem also gives a lower bound L^* on the optimal objective function value of (P). We now show that $z^\circ \leq L^*$; that is, Lagrangian relaxation yields a lower bound that is at least as good as that obtained from the linear programming relaxation. We establish this result by showing that the Lagrangian multiplier problem also solves a linear programming problem but that the solution space for this problem is contained within the solution space of the problem (LP). The linear programming problem that the Lagrangian multiplier problem solves uses "convexification" of the solution space $X = \{x : \mathcal{D}x \leq q, x \geq 0 \text{ and integer}\}$.

We assume that $X = \{x^1, x^2, \ldots, x^K\}$ is a finite set. We say that a solution x is a *convex combination* of the solutions $x^1, x^2, \ldots, x^K$ if $x = \sum_{k=1}^K \lambda_k x^k$ for some nonnegative weights $\lambda_1, \lambda_2, \ldots, \lambda_K$ satisfying the condition $\sum_{k=1}^K \lambda_k = 1$. Let $\mathcal{H}(X)$ denote the *convex hull* of X (i.e., the set of all convex combinations of X). In the subsequent discussion we use the following properties of $\mathcal{H}(X)$.

Property 16.7

(a) The set $\mathcal{H}(X)$ is a polyhedron, that is, it can be expressed as a solution space defined by a finite number of linear inequalities.

(b) Each extreme point solution of the polyhedron $\mathcal{H}(X)$ lies in X, and if we optimize a linear objective function over $\mathcal{H}(X)$, some solution in X will be an optimal solution.

(c) The set $\mathcal{H}(X)$ is contained in the set of solutions $\{x : \mathcal{D}x \leq q, x \geq 0\}$.

Proof. Part (a) is a well-known result in linear algebra which we do not prove. The first statement in part (b) follows from the fact that every point of $\mathcal{H}(X)$ not in X is a convex combination, with positive weights, of two or more points in X and so is not an extreme point (see Appendix C). The second statement in part (b) is a consequence of the fact that linear programs always have at least one extreme point solution (see Appendix C). Part (c) follows from the fact that every solution in X also belongs to the convex set $\{x : \mathcal{D}x \leq q, x \geq 0\}$, and consequently, every convex combination of solutions in X, which defines $\mathcal{H}(X)$, also belongs to the set $\{x : \mathcal{D}x \leq q, x \geq 0\}$. ◆

We now prove the main result of this section.

Theorem 16.8. *The optimal objective function value L^* of the Lagrangian multiplier problem equals the optimal objective function value of the linear program $\min\{cx : \mathcal{A}x = b, x \in \mathcal{H}(X)\}$.*

Proof. Consider the Lagrangian subproblem

$$L(\mu) = \min\{cx + \mu(\mathcal{A}x - b) : x \in X\},$$

for some choice μ of the Lagrange multipliers. This problem is equivalent to the problem

$$L(\mu) = \min\{cx + \mu(\mathcal{A}x - b) : x \in \mathcal{H}(X)\}, \tag{16.2}$$

because by Property 16.7(b), some extreme point solution of $\mathcal{H}(X)$ solves this problem and each extreme point solution of $\mathcal{H}(X)$ belongs to X. Now, notice that the Lagrangian subproblem defined by (16.2) is a linear programming problem because by Property 16.7(a), we can formulate the set $\mathcal{H}(X)$ as the set of solutions of a finite number of linear inequalities. Therefore, we can conceive of the Lagrangian subproblem (16.2) as a relaxation of the following linear programming problem:

$$\min\{cx : \mathcal{A}x = b, x \in \mathcal{H}(X)\}.$$

Finally, we use Theorem 16.6 to observe that the optimal value L^* of the Lagrangian multiplier problem equals the optimal objective function value of the linear program $\min\{cx : \mathcal{A}x = b, x \in \mathcal{H}(X)\}$. ◆

We subsequently refer to the problem $\min\{cx : \mathcal{A}x = b, x \in \mathcal{H}(X)\}$ as the *convexified version* of problem (P) and refer to it as (CP). The preceding theorem shows that L^* equals the optimal objective function value of the convexified problem. What is the relationship between the set of feasible solutions of the convexified problem (CP) and the linear programming relaxation (LP)? We illustrate this relationship using a numerical example.

For simplicity, in our example we assume that the relaxed constraints are of the form $\mathcal{A}x \leq b$ instead of $\mathcal{A}x = b$. We consider a two-variable problem with the constraints $\mathcal{A}x \leq b$ and $\mathcal{D}x \leq q$ as shown in Figure 16.6(a). This figure also specifies the set of solutions of the integer programming problem (P), denoted by the circled points. Figure 16.6(b) shows the solution space of the linear programming relaxation (LP) of the problem. Figure 16.6(c) shows the convex hull $\mathcal{H}(X)$ and Figure 16.6(d) depicts the solution space of the convexified problem (CP). Note that the solution space of (CP) is a subset of the solution space of (LP).

The preceding result is also easy to establish in general. Notice from Property 16.7(c) that since $\mathcal{H}(X)$ is contained in the set $\{x : \mathcal{D}x \leq q, x \geq 0\}$, the set of solutions of problem (CP) given by $\{x : \mathcal{A}x = b, x \in \mathcal{H}(X)\}$ is contained in the set of solutions of (LP) given by $\{x : \mathcal{A}x = b, \mathcal{D}x \leq q, x \geq 0\}$. Since optimizing the same objective function over a smaller solution space cannot improve the objective function value, we see that $z^\circ \leq L^*$. We state this important result as a theorem.

Theorem 16.9. *When applied to an integer program stated in minimization form, the lower bound obtained by the Lagrangian relaxation technique is always as large (or, sharp) as the bound obtained by the linear programming relaxation of the problem; that is,* $z^\circ \leq L^*$.

Under what situations will the Lagrangian bound equal the linear programming

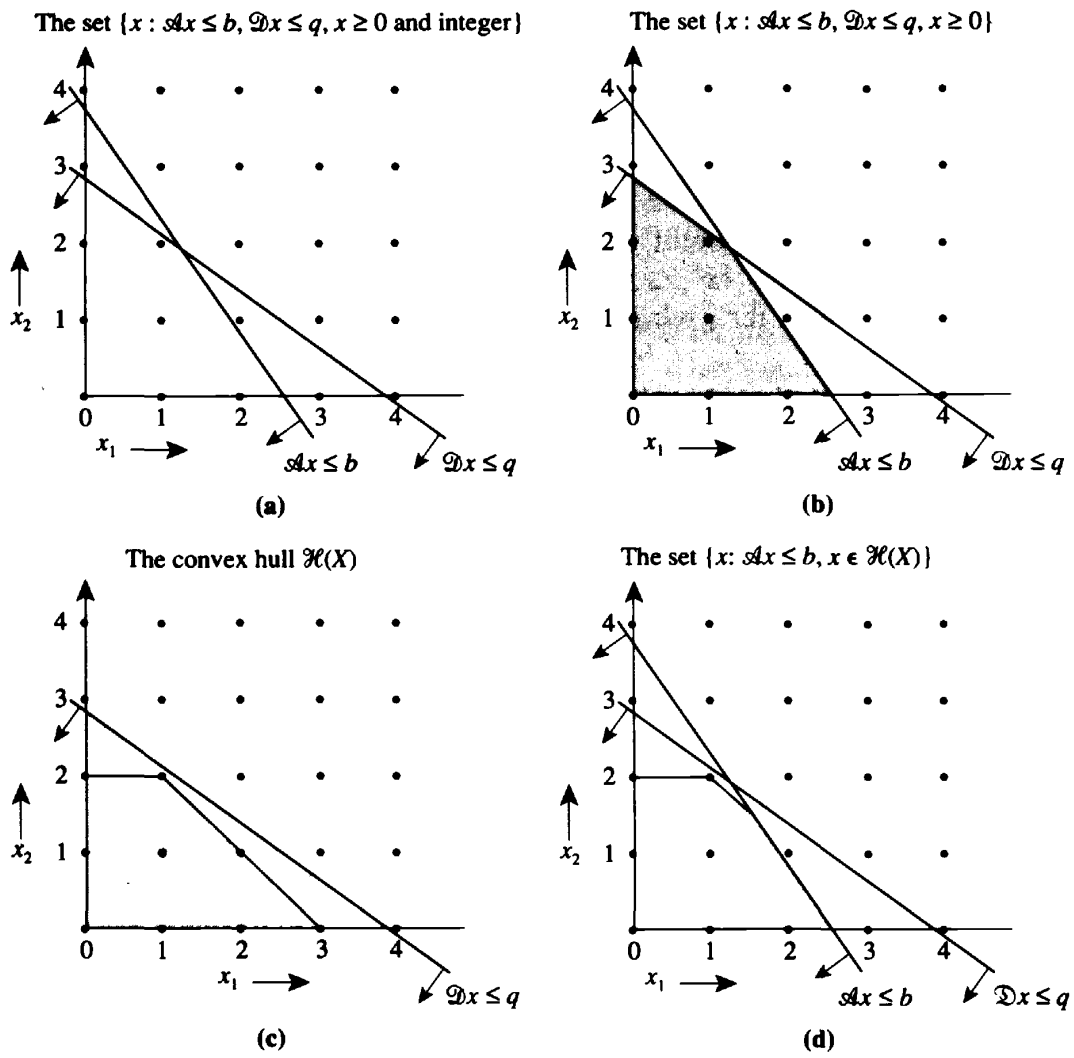

Figure 16.6 Illustrating the relationship between the problem (LP) and (CP): (a) solution space of the integer program (P); (b) solution space of the linear programming relaxation (LP); (c) convex hull $\mathcal{H}(x)$; (d) solution space of the convexified problem (CP).

bound? We show that if the Lagrangian subproblem satisfies a property, known as the *integrality property*, the Lagrangian bound will equal the linear programming bound. We say that the Lagrangian subproblem $\min\{dx : \mathcal{D}x \le q, x \ge 0$ and integer$\}$ satisfies the integrality property if it has an integer optimal solution for every choice of objective function coefficients *even if we relax the integrality restrictions on the variables* x. Note that this condition implies that the problems $\min\{cx + \mu(\mathcal{A}x - b) : \mathcal{D}x \le q, x \ge 0$ and integer$\}$ and $\min\{cx + \mu(\mathcal{A}x - b) : \mathcal{D}x \le q, x \ge 0\}$ have the same optimal objective function values for every choice of the Lagrange multiplier μ. For example, if the constraints $\mathcal{D}x \le q$ are the mass balance constraints of a minimum cost flow problem (or any of its special cases, such as the maximum flow, shortest path, and assignment problems), the problem $\min\{cx + \mu(\mathcal{A}x - b) : \mathcal{D}x \le q, x \ge 0\}$ will always have an integer optimal solution and imposing integrality constraints on the variables will not increase the optimal objective function value.

Theorem 16.10. *If the Lagrangian subproblem of the optimization problem* (P) *satisfies the integrality property, then* $z^\circ = L^*$.

Proof. Observe that the problem $\min\{dx : \mathcal{D}x \leq q, x \geq 0\}$ will have an integer optimal solution for every choice of d only if every extreme point solution of the constraints $\mathcal{D}x \leq q$, $x \geq 0$, is integer; for otherwise, we can select d so that a noninteger extreme point solution becomes an optimal solution. This observation implies that the set $\{x : \mathcal{D}x \leq q, x \geq 0\}$ equals the convex hull of $X = \{x : \mathcal{D}x \leq q, x \geq 0 \text{ and integer}\}$, which we have denoted by $\mathcal{H}(X)$. This result further implies that the sets $\{x : \mathcal{A}x = b, \mathcal{D}x \leq q, x \geq 0\}$ and $\{x : \mathcal{A}x = b, x \in \mathcal{H}(X)\}$ are the same. The first of these sets is the set of feasible solutions of the linear programming relaxation (LP) and the latter set is the set of feasible solutions of the convexified problem (CP). Since both the problems (LP) and (CP) have the same set of feasible solutions, they will have the same optimal objective function value, which is the desired conclusion of the theorem. ◆

This result shows that for problems satisfying the integrality property, solving the Lagrangian multiplier problem is equivalent to solving the linear programming relaxation of the problem. In these situations the Lagrangian relaxation technique provides no better a bound than the linear programming relaxation. Nevertheless, the Lagrangian relaxation technique might still be of considerable value, because solving the Lagrangian multiplier problem might be more efficient than solving the linear programming relaxation directly. Network optimization problems perhaps provide the most useful problem domain for exploiting this result because the Lagrangian subproblem in these cases often happens to be a minimum cost flow problem or one of its specializations.

As we have noted previously, in many (in fact, most) problem instances, the optimal objective function value L^* of the Lagrangian multiplier problem will be strictly less than the optimal objective function value z^* of problem (P); that is, the problem has a *duality gap*. As an example, consider the constrained shortest path example that we discussed in Section 16.3. For this example, $L^* = 7$ and $z^* = 13$. The duality gap occurs because the Lagrangian multiplier problem solves an optimization problem over a larger solution space (its convexification) than that of the original problem (P), and consequently, its optimal objective function value might be smaller.

16.5 APPLICATIONS OF LAGRANGIAN RELAXATION

As we noted earlier in the chapter, Lagrangian relaxation has many applications in network optimization. In this section we illustrate the breadth of these applications. The selected applications are both important in practice and illustrate how many of the network models we have considered in earlier chapters arise as Lagrangian subproblems. We consider the following models with embedded network structure.

Topic	Embedded network structure
Networks with side constraints	• Minimum cost flows • Shortest paths
Traveling salesman problem	• Assignment problem • Minimum cost flows
Vehicle routing	• Assignment problem • A variant of minimum spanning tree
Network design	• Shortest paths
Two-duty operator scheduling	• Shortest paths • Minimum cost flows
Degree-constrained minimum spanning trees	• Minimum spanning tree
Multi-item production planning	• Shortest paths • Minimum cost flows • Dynamic programs

Application 16.1 Networks with Side Constraints

The constrained shortest path problem is a special case of a broader set of optimization models known as network flow problems with side constraints. We can formulate a generic version of this problem as follows:

$$\text{Minimize} \quad cx$$

subject to

$$\mathcal{A}x \leq b,$$

$$\mathcal{N}x = q,$$

$$l \leq x \leq u, \text{ and } x_{ij} \text{ integer} \qquad \text{for all } (i, j) \in I.$$

In this formulation, as in the usual minimum cost flow problem, x is a vector of arc flows, $\mathcal{N}$ is a node–arc incidence matrix, q is a vector of node supplies and demands, and l and u are lower and upper bounds imposed on the arc flows. The set I is an index set of variables that must be integer. The flow vector x might be constrained to be integer or not, depending on the application being modeled. The added complication in this model are the side constraints $\mathcal{A}x \leq b$ that further restrict the arc flows.

For example, in the constrained shortest path problem, the network constraints model a shortest path problem [i.e., $q(s) = 1$ and $q(t) = -1$ for the source node s and destination node t, and $q(j) = 0$ for every other node j; also, every lower bound $l_{ij} = 0$ and every upper bound $u_{ij} = \infty$]. In this case the side constraint $\sum_{(i,j)\in A} t_{ij}x_{ij} \leq T$ is a single inequality constraint modeling the timing restriction.

The network flow model with side constraints arises in many application contexts in which the arc flows consume scarce resources (e.g., labor) or we wish to impose service constraints on the flows (e.g., maximum delay times in a communication and transportation network). The model also arises when the network flow model has multiple commodities, each governed by their own flow constraints, that

share common resources such as arc capacities. In Chapter 17 we consider one such model, the classical multicommodity flow problem, in some detail.

We might note that the network flow model with side constraints also arises in other, perhaps more surprising ways. As an illustration, consider a standard work force scheduling problem. Suppose that we wish to schedule employees (e.g., telephone operators, production workers, or nurses) in a way that ensures that $\alpha(j)$ employees are available for work on the jth day of the week; suppose, further, that we wish to schedule the employees so that each has two consecutive days off each week. That is, each of them works 5 consecutive days and then has 2 days off. We incur a cost c_j for each employee that is scheduled to work on day j. Figure 16.7 shows a network flow model with side constraints for this problem. The network contains three types of arcs.

1. A "work arc" for each day of the week: The flow on this arc is the number of employees scheduled to work on that day; the arc has an associated cost (e.g., weekends might have a pay premium and so a higher cost) and a lower flow bound equaling the number of employees required to work on that day.
2. A "total work force arc" that introduces the work force at the beginning of the planning cycle (which we arbitrarily take to be Sunday) and removes it at the end of the planning cycle (Saturday): the flow y on this arc is the total number of employees employed during the week.
3. "Days-off arcs" with the flows $x_{\text{sun}}, x_{\text{mon}}, \ldots, x_{\text{sat}}$, each representing a schedule with 2 days off beginning with the day indicated by the subscript: The flow on arc x_{sun}, for example, bypasses the Sunday and Monday work arcs, indicating that the employees working in this schedule are not available for work on Sunday and Monday.

A complicating feature of this network flow model is a single additional constraint indicating that every employee must be assigned to at least one schedule; that is,

$$y = x_{\text{sun}} + x_{\text{mon}} + \cdots + x_{\text{sat}}.$$

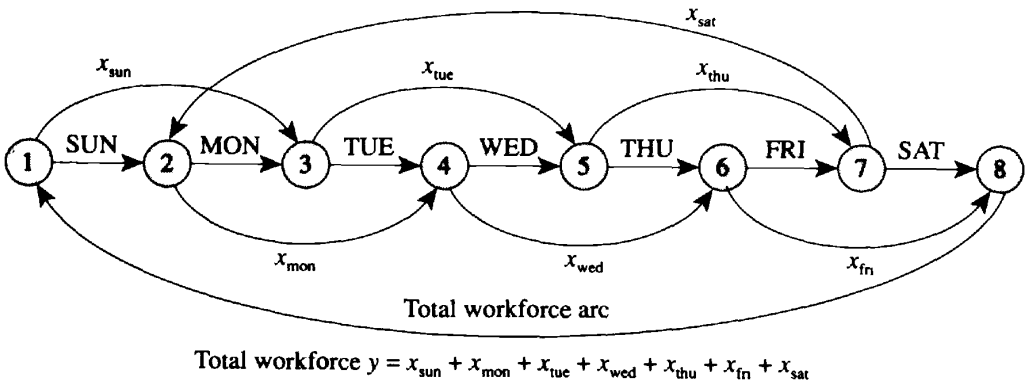

Figure 16.7 Network model of the cyclic scheduling problem. Lower bound on day arcs = demand for the day.

This side constraint specifies a flow relationship between several of the arcs in the network flow model. Relaxing this constraint and using Lagrangian relaxation provides us with one algorithmic approach for solving this problem. The algorithmic procedure for applying Lagrangian relaxation to the general network flow model with side constraints is essentially the same as the procedure we have discussed for the constrained shortest path problem: we associate nonnegative Lagrange multipliers μ with the side constraints $\mathcal{A}x \leq b$ and bring them into the objective function to produce the network flow subproblem

$$\text{minimize}\{cx + \mu(\mathcal{A}x - b) : \mathcal{N}x = q, l \leq x \leq u\},$$

and then solve a sequence of these problems with different values of the Lagrange multipliers μ which we update using the subgradient optimization technique. For each choice of the Lagrangian multiplier on this constraint, the Lagrangian subproblem is a network flow problem. In Exercise 9.9 we show that we can actually solve this special case of network flows with side constraints much more efficiently by solving a polynomial sequence of network flow problems.

Application 16.2 Traveling Salesman Problem

The traveling salesman problem is perhaps the most famous problem in all of network and combinatorial optimization: Its simplicity and yet its difficulty have made it an alluring problem that has attracted the attention of many noted researchers over a period of several decades. The problem is deceptively easy to state: Starting from his home base, node 1, a salesman wishes to visit each of several cities, represented by nodes 2, . . . , n, exactly once and return home, doing so at the lowest possible travel cost. We will refer to any feasible solution to this problem as a *tour* (of the cities).

The traveling salesman problem is a generic core model that captures the combinatorial essence of most routing problems and, indeed, most other routing problems are extensions of it. For example, in the classical vehicle routing problem, a set of vehicles, each with a fixed capacity, must visit a set of customers (e.g., grocery stores) to deliver (or pick up) a set of goods. We wish to determine the best possible set of delivery routes. Once we have assigned a set of customers to a vehicle, that vehicle should take the minimum cost tour through the set of customers assigned to it; that is, it should visit these customers along an optimal traveling salesman tour.

The traveling salesman problem also arises in problems that on the surface have no connection with routing. For example, suppose that we wish to find a sequence for loading jobs on a machine (e.g., items to be painted), and that whenever the machine processes job i after job j, we must reset the machine (e.g., clear the dies of the colors of the previous job), incurring a setup time c_{ij}. Then in order to find the processing sequence that minimizes the total setup time, we need to solve a traveling salesman problem—the machine, which functions as the "salesman," needs to "visit" the jobs in the most cost-effective manner.

There are many ways to formulate the traveling salesman problem as an optimization model. We present a model with an embedded (directed) network flow structure. Exercises 16.21 and 16.23 consider other modeling approaches. Let c_{ij}

denote the cost of traveling from city i to city j and let y_{ij} be a zero–one variable, indicating whether or not the salesman travels from city i to city j. Moreover, let us define flow variables x_{ij} on each arc (i, j) and assume that the salesman has $n - 1$ units available at node 1, which we arbitrarily select as a "source node," and that he must deliver 1 unit to each of the other nodes. Then the model is

$$\text{Minimize} \quad \sum_{(i,j)\in A} c_{ij}y_{ij} \tag{16.3a}$$

subject to

$$\sum_{1\le j\le n} y_{ij} = 1 \qquad \text{for all } i = 1, 2, \ldots, n, \tag{16.3b}$$

$$\sum_{1\le i\le n} y_{ij} = 1 \qquad \text{for all } j = 1, 2, \ldots, n, \tag{16.3c}$$

$$\mathcal{N}x = b, \tag{16.3d}$$

$$x_{ij} \le (n - 1)y_{ij} \qquad \text{for all } (i, j) \in A, \tag{16.3e}$$

$$x_{ij} \ge 0 \qquad \text{for all } (i, j) \in A, \tag{16.3f}$$

$$y_{ij} = 0 \text{ or } 1 \qquad \text{for all } (i, j) \in A. \tag{16.3g}$$

To interpret this formulation, let $A' = \{(i, j) : y_{ij} = 1\}$ and let $A'' = \{(i, j) : x_{ij} > 0\}$. The constraints (16.3b) and (16.3c) imply that exactly one arc of A' leaves and enters any node i; therefore, A' is the union of node disjoint cycles containing all of the nodes of N. In general, any integer solution satisfying (16.3b) and (16.3c) will be the union of disjoint cycles; if any such solution contains more than one cycle, we refer to each of the cycles as subtours, since they pass through only a subset of the nodes. Figure 16.8 gives an example of a subtour solution to the constraints (16.3b) and (16.3c).

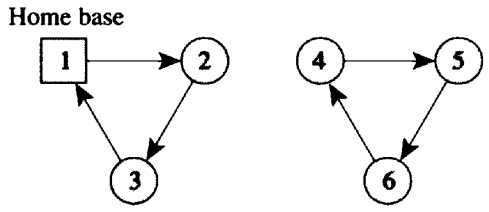

Figure 16.8 Infeasible solution for the traveling salesman problem containing subtours.

Constraint (16.3d) ensures that A'' is connected since we need to send 1 unit of flow from node 1 to every other node via arcs in A''. The "forcing" constraints (16.3e) imply that A'' is a subset of A'. [Notice that since no arc need ever carry more than $(n - 1)$ units of flow, the forcing constraint for arc (i, j) is redundant if $y_{ij} = 1$.] These conditions imply that the arc set A' is connected and so cannot contain any subtours. We conclude that the formulation (16.3) is a valid formulation for the traveling salesman problem.

One of the nice features of this formulation is that we can apply Lagrangian relaxation to it in several ways. For example, suppose that we attach Lagrange multipliers $\mu_{ij} \ge 0$ with the forcing constraints (16.3e) and bring them into the objective function, giving the Lagrangian objective function

$$\text{Minimize} \sum_{(i,j)\in A} [c_{ij} - (n-1)\mu_{ij}]y_{ij} + \sum_{(i,j)\in A} \mu_{ij}x_{ij},$$

and leaving (16.3b)–(16.3d), (16.3f), and (16.3g) as constraints in the Lagrangian subproblem. Note that nothing in this Lagrangian subproblem couples the variables y_{ij} and x_{ij}. Therefore, the subproblem decomposes into two separate subproblems: (1) an assignment problem in the variables y_{ij}, and (2) a minimum cost flow problem in the variables x_{ij}. So for any choice of the Lagrangian multipliers μ, we solve two network flow subproblems; by using subgradient optimization we can find the best lower bound and optimal values of the multipliers. By relaxing other constraints in this model, or by applying Lagrangian relaxation to other formulations of the traveling salesman problem, we could define other network flow subproblems (see Exercise 16.19).

Application 16.3 Vehicle Routing

The vehicle routing problem is a generic model that practitioners encounter in many problem settings including the delivery of consumer products to grocery stores, the collection of money from vending machines and telephone coin boxes, and the delivery of heating oil to households. As we have noted earlier in this section, the vehicle routing problem is a generalization of the traveling salesman problem.

The vehicle routing problem is easy to state: Given (1) a fleet of K capacitated vehicles domiciled at a common depot, say node 1, (2) a set of customer sites $j = 2, 3, \ldots, n$, each with a prescribed demand d_j, and (3) a cost c_{ij} of traveling from location i to location j, what is the minimum cost set of routes for delivering (picking up) the goods to the customer sites? We assume that the vehicle fleet is homogeneous and that each vehicle has a capacity of u units.

There are many different variants on this core vehicle routing problem. For example, the vehicle fleet might be nonhomogeneous, each vehicle route might have a total travel time restriction, or deliveries for each customer might have time window restrictions (earliest and latest delivery times). We illustrate the use of Lagrangian relaxation by considering only the basic model, which we formulate with decision variables x_{ij}^k indicating whether ($x_{ij}^k = 1$) or not ($x_{ij}^k = 0$) we dispatch vehicle k on arc (i, j) and y_{ij} indicating whether some vehicle travels on arc (i, j):

$$\text{Minimize} \sum_{1\le k\le K} \sum_{(i,j)\in A} c_{ij}x_{ij}^k \tag{16.4a}$$

subject to

$$\sum_{1\le k\le K} x_{ij}^k = y_{ij}, \tag{16.4b}$$

$$\sum_{1\le j\le n} y_{ij} = 1 \qquad \text{for } i = 2, 3, \ldots, n, \tag{16.4c}$$

$$\sum_{1\le i\le n} y_{ij} = 1 \qquad \text{for } j = 2, 3, \ldots, n, \tag{16.4d}$$

$$\sum_{1\le j\le n} y_{1j} = K, \tag{16.4e}$$

$$\sum_{1\le i\le n} y_{i1} = K, \tag{16.4f}$$

$$\sum_{2 \le i \le n} \sum_{1 \le j \le n} d_i x_{ij}^k \le u \qquad \text{for all } k = 1, 2, \ldots, K, \tag{16.4g}$$

$$\sum_{i \in Q} \sum_{j \in Q} y_{ij} \le |Q| - 1 \qquad \text{for all subsets } Q \text{ of } \{2, 3, \ldots, n\}, \tag{16.4h}$$

$$y_{ij} = 0 \text{ or } 1 \qquad \text{for all } (i,j) \in A, \tag{16.4i}$$

$$x_{ij}^k = 0 \text{ or } 1 \qquad \text{for all } (i,j) \in A \text{ and all } k = 1, 2, \ldots, K. \tag{16.4j}$$

Let $A' = \{(i, j) : y_{ij} = 1\}$. As in our discussion of the traveling salesman problem, constraints (16.4c) and (16.4d) ensure that A' is the union of node disjoint cycles containing all of the nodes in N. Constraint (16.4h) ensures that the solution must contain no cycle using the nodes 2, 3, . . . , n (i.e., not contain any subtours on these nodes); otherwise, the arcs A' would contain some cycle passing through a set Q of nodes and the solution would violate constraint (16.4h) since the left-hand side of the constraint (16.4h) would be at least $|Q|$. For this reason, we refer to the constraints (16.4h) as *subtour breaking constraints.*

We might note that if $K = 1$, and u is so large that the constraint (16.4g) is redundant, this model becomes an "assignment-based" formulation of the traveling salesman problem, which is an alternative formulation to the "flow-based" model that we introduced previously as (16.3). In Exercise 16.24 we study the relationships between these formulations as well as a third model, a multicommodity flow-based formulation.

Note that this formulation has several embedded structures that we might exploit in a Lagrangian relaxation solution approach. By relaxing some of the constraints, we are also able to decompose the problem into independent subproblems. For example, if we relax only constraints (16.4b), no constraint connects the x variables and y variables, so the problem decomposes into separate subproblems in each of these variables. By relaxing different combinations of the constraints, we create several different types of subproblems:

1. If we relax the constraints (16.4b), (16.4g), and (16.4h), the resulting formulation is an assignment problem.
2. If we relax the constraints (16.4b) to (16.4f), and (16.4h), the resulting problem decomposes into independent "knapsack problems," one for each vehicle k.
3. If we relax constraint (16.4b), the problem decomposes into separate subproblems, one in the y variables and one in the x^k variables for each vehicle k. The first of these problems is a so-called K-traveling salesman problem (see Exercise 16.25) and each problem in the variables x^k is a knapsack problem.
4. If we relax the assignment constraints (16.4c) to (16.4f), the constraint (16.4b) defining y, and the capacity constraint (16.4g), the resulting problem is a minimum forest problem on the nodes 2, 3, . . . , n. This problem is easy to solve by a simple variant of any minimum spanning tree algorithm. We could strengthen this approach by adding other (redundant) constraints to the problem formulation (see Exercise 16.28).
5. If we relax constraints (16.4b), (16.4c), and (16.4e) to (16.4g), the subproblem with the constraints (16.4d), (16.4h), and (16.4i) becomes a directed minimum spanning tree problem—any feasible solution will be a directed spanning tree

with exactly one arc directed into each node (except for the root node 1). Although we do not consider this problem in this book, it is polynomially solvable.

6. If we relax the constraint (16.4g), the problem becomes a variant of the K-traveling salesman problem.

These various possibilities illustrate the remarkable flexibility of the Lagrangian relaxation solution approach.

Application 16.4 Network Design

Suppose that we have the flexibility of designing a network as well as determining its optimal flow (routing). That is, we have a directed network $G = (N, A)$ and can introduce an arc or not into the design of the network: If we use (introduce) an arc (i, j), we incur a design (construction) cost f_{ij}. Our problem is to find the design that minimizes the total systems cost—that is, the sum of the design cost and the routing cost. This type of model arises in many application contexts, for example, the design of telecommunication or computer networks, load planning in the trucking industry (i.e., the design of a routing plan for trucks), and the design of production schedules.

Many alternative modeling assumptions arise in practice. We consider one version of the problem, the *uncapacitated network design problem*. In this model we need to route multiple commodities on the network; each commodity k has a single source node s^k and a single destination node d^k. Once we introduce an arc (i, j) into the network, we have sufficient capacity to route all of the flow by all commodities on this arc.

To formulate this problem as an optimization model, let x^k denote the vector of flows of commodity k on the network. Rather than letting x_{ij}^k model the total flow of commodity k on arc (i, j), however, we let x_{ij}^k denote the fraction of the required flow of commodity k to be routed from the source s^k to the destination d^k that flows on arc (i, j). Let c^k denote the cost vector for commodity k, which we scale to reflect the way that we have defined x_{ij}^k [i.e., c_{ij}^k is the per unit cost for commodity k on arc (i, j) times the flow requirement of that commodity]. Also, let y_{ij} be a zero–one vector indicating whether or not we select arc (i, j) as part of the network design. Using this notation, we can formulate the network design problem as follows:

$$\text{Minimize} \quad \sum_{1 \le k \le K} c^k x^k + fy \tag{16.5a}$$

subject to

$$\sum_{\{j:(i,j)\in A\}} x_{ij}^k - \sum_{\{j:(j,i)\in A\}} x_{ji}^k = \begin{cases} 1 & \text{if } i = s^k \\ -1 & \text{if } i = d^k \\ 0 & \text{otherwise} \end{cases} \quad \text{for all } i \in N,\ k = 1, 2, \ldots, K, \tag{16.5b}$$

$$x_{ij}^k \le y_{ij} \qquad \text{for all } (i, j) \in A,\ k = 1, 2, \ldots, K, \tag{16.5c}$$

$$x_{ij}^k \geq 0 \qquad \text{for all } (i, j) \in A \text{ and all } k = 1, 2, \ldots, K, \tag{16.5d}$$

$$y_{ij} = 0 \text{ or } 1 \qquad \text{for all } (i, j) \in A. \tag{16.5e}$$

In this formulation, the "forcing constraints" (16.5c) state that if we do not select arc (i, j) as part of the design, we cannot flow any fraction of commodity k's demand on this arc, and if we do select arc (i, j) as part of the design, we can flow as much of the demand of commodity k as we like on this arc.

Note that if we remove the forcing constraints from this model, the resulting model in the flow variables x^k decomposes into a set of independent shortest path problems, one for each commodity k. Consequently, the model is another attractive candidate for the application of Lagrangian relaxation. To see why this type of solution approach might be attractive, consider a typically sized problem with, say, 50 nodes and 500 candidate arcs. Suppose that we have a separate commodity for each pair of nodes (as is typical in communication settings in which each node is sending messages to every other node). Then we have 50(49) = 2450 commodities. Since each commodity can flow on each arc, the model has 2450(500) = 1,225,000 flow variables, and since (1) each flow variable defines a forcing constraint, and (2) each commodity has a flow balance constraint at each node, the model has 1,225,000 + 2450(50) = 1,347,500 constraints. In addition, it has 500 zero–one variables. So even as a linear program, this model far exceeds the capabilities of current state of the art software systems. By decomposing the problem, however, for each choice of the vector of Lagrange multipliers, we will solve 2450 small shortest path problems.

Application 16.5 Two-Duty Operator Scheduling

In many different problem contexts in work force planning, a private firm or public-sector organization must schedule its employees—for example, nurses, airline crews, telephone operators—to provide needed services. Typically, the problems are complicated by complex work rules, for example, airline crews have limits on the number of hours that they can fly in any week or month. Moreover, frequently, the demand for the services of these employees varies considerably by time of the day or week, or across geography (as in the case of airline crew scheduling). Consequently, finding a minimum cost schedule requires that we balance the prevailing work rules with the demand patterns. Figure 16.9 shows one example of a work force planning problem, which we will view as a driver schedule for a single bus line.

Every column in this table corresponds to a possible schedule. For example, in schedule 1, a driver operates the bus line in two shifts, from 8 to 11 and then from 1 to 3; in schedule 2 the driver works a single shift, from 11 to 1, and in schedule 3, he/she drives from 3 to 6. As indicated by the column entitled demand in the table, we wish to find a set of schedules satisfying the property that at least one driver is assigned to the bus at every hour of the day from 8 A.M. until 6 P.M. (if two drivers are assigned to the same bus at the same time, one drives and the other is a rider). One possibility is to choose schedules 1, 2, and 3; another is schedules 4 and 5; and still another is schedules 3, 5, and 6. Each schedule j has an associated

Time period	Schedule								Demand
	1	2	3	4	5	6	7	8	
8–9	1	0	0	1	0	1	0	1	≥ 1
9–10	1	0	0	1	0	1	0	1	≥ 1
10–11	1	0	0	0	1	0	0	1	≥ 1
11–12	0	1	0	0	1	0	0	1	≥ 1
12–1	0	1	0	0	1	0	0	1	≥ 1
1–2	1	0	0	1	0	1	1	0	≥ 1
2–3	1	0	0	1	0	1	1	0	≥ 1
3–4	0	0	1	1	0	0	1	0	≥ 1
4–5	0	0	1	1	0	0	1	0	≥ 1
5–6	0	0	1	1	0	0	1	0	≥ 1
Cost	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	

Figure 16.9 Two-duty operator schedule.

cost c_j and we wish to choose the set of schedules that meets the scheduling requirement at the lowest possible cost. To formulate this problem formally as an optimization model, let x_j be a binary (i.e., zero–one) variable indicating whether ($x_j = 1$) or not ($x_j = 0$), we choose schedule j, and let $\mathcal{A}$ denote the zero–one matrix of coefficients of the scheduling table (i.e., the ijth element is 1 if schedule j has a driver on duty during the ith hour of the day). Also, let e denote a column of 1's. Then the model is

$$\text{Minimize} \quad cx \tag{16.6a}$$

subject to

$$\mathcal{A}x \geq e, \tag{16.6b}$$

$$x_j = 0 \text{ or } 1 \qquad \text{for } j = 1, 2, \ldots, n. \tag{16.6c}$$

The choice of the available schedules in the problem depends on the governing work rules; as an illustration, in our example, no operator works in any shift of less than 2 hours. Moreover, note that the schedules permit split shifts, that is, time on, time off, and then time on again as in schedule 1. Note, however, that no schedule has more than two shifts. We refer to this special version of the general operator scheduling problem as the *two-duty operator scheduling problem.*

In Exercise 4.13 we showed how to solve the single-duty scheduling problem as a shortest path problem: The shortest path model contains a node for each time period $1, 2, \ldots, T$ to be covered, plus an artificial end node $T + 1$, and an arc from node i to node j whenever a schedule starts at the beginning of time period i and ends at the beginning of time period j. We interpret arc (i, j) as "covering" the time periods $i, i + 1, \ldots, j - 1$. The network also contains "backward" arcs of the form $(j + 1, j)$ that permits us to "back up" from time period $j + 1$ to time period j so that we can cover any node more than once and model the possibility that a schedule might assign more than one driver to any time period. Can we use

Lagrangian relaxation to exploit the fact that the single-duty problem is a shortest path problem? To do so, we will use an idea known as *variable splitting*.

Consider any column j of the matrix $\mathcal{A}$ that contains two sequences of 1's—that is, corresponds to a schedule with two shifts. Let us make two columns $\mathcal{A}'_j$ and $\mathcal{A}''_j$ out of this column; each of these columns contains one of the duties (sequence of 1's) from $\mathcal{A}_j$, so $\mathcal{A}_j = \mathcal{A}'_j + \mathcal{A}''_j$. Let us also replace the variable x_j in our model with two variables x'_j and x''_j. We form a new model with these variables as

$$\text{Minimize} \quad c'x' + c''x'' \tag{16.7a}$$

subject to

$$\mathcal{A}'x' + \mathcal{A}''x'' \geq e, \tag{16.7b}$$

$$x' - x'' = 0, \tag{16.7c}$$

$$x'_j \text{ and } x''_j = 0 \text{ or } 1 \qquad \text{for } j = 1, 2, \ldots, n. \tag{16.7d}$$

For convenience, in formulating this model we have assumed that we have split every column of the matrix $\mathcal{A}$. If not, we can simply assume that some columns of $\mathcal{A}''$ are columns of zeros. Moreover, we can split the cost of each variable x_j arbitrarily between c'_j and c''_j. For example, we could let each of these costs be half of c_j. This model and the original model (16.6) are clearly equivalent. Note, however, that the new model reveals embedded network structure; as shown in Figure 16.10, which is the network model associated with the data in Figure 16.9, the model is a shortest path problem with the complicating constraints that we need to choose arcs in pairs: We choose either both or none of the arcs corresponding to the variables x'_j and x''_j. If we eliminate the "complicating" constraint $x' - x'' = 0$, the problem becomes an easily solvable single duty scheduling problem, which as we have seen before, we can solve via a shortest path computation. This observation suggests that we adopt a Lagrangian relaxation approach, relaxing the constraints with a Lagrange multiplier μ so that the Lagrangian subproblem has the objective function

$$L(\mu) = \min(c' + \mu)x' + (c'' - \mu)x''. \tag{16.8}$$

Now, as usual to solve the Lagrangian multiplier problem, we apply subgradient

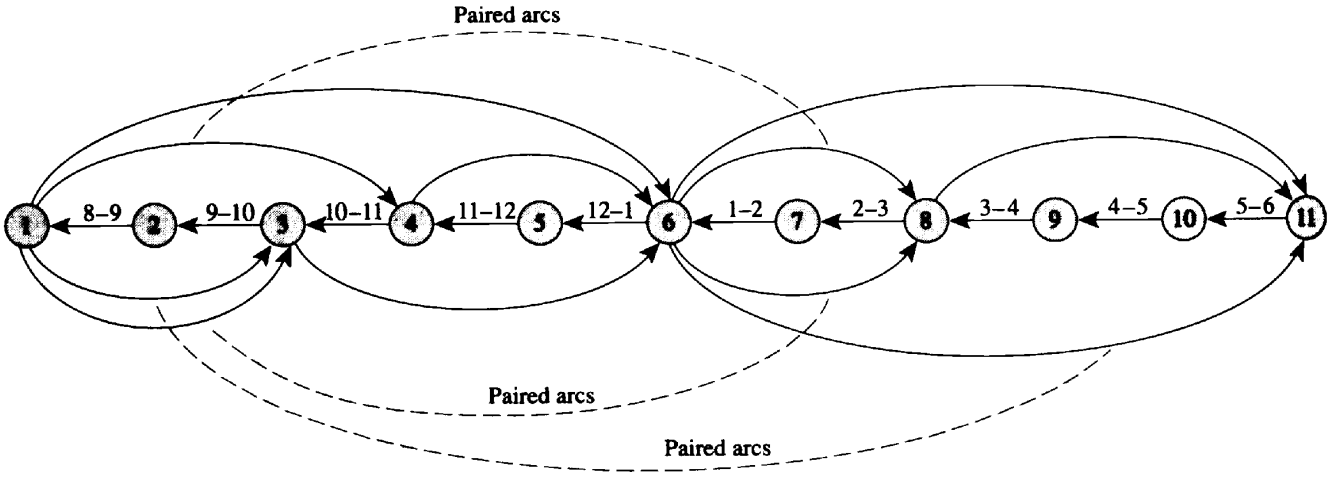

Figure 16.10 Shortest path subproblem for the two duty scheduling problem.

optimization, or some other solution technique, to maximize $L(\mu)$ over all possible choices of the Lagrange multipliers μ.

When we split a column A_j into two columns A_j' and A_j'', it does not matter how we split the cost c_j between c_j' and c_j'', as long as $c_j = c_j' + c_j''$. Since $x_j' = x_j''$ in any feasible solution, the cost of any feasible solution will be the same no matter how we allocate the cost. However, cost splitting does make a significant difference in the relaxed problem obtained by dropping the constraint $x_j' = x_j''$. If we were to make c_j' large and c_j'' small, then in the solution to the relaxed problem we would probably find that x_j' would be 0 and x_j'' would be 1. Similarly, if we made c_j' small and c_j'' large, in the solution to the relaxed problem we would likely find that x_j' would be 1 and x_j'' would be 0. Ideally, we should allocate the costs between c_j' and c_j'', so that either $x_j' = x_j'' = 0$ or $x_j' = x_j'' = 1$ in the relaxed problem.

As it turns out, we need not worry about the cost allocation at all if we use Lagrangian relaxation since the Lagrange multiplier μ_j for the constraint $x_j' - x_j'' = 0$ does the cost allocation. Suppose, for example, that $c_j' = c_j$ and $c_j'' = 0$. Then, since we are relaxing the constraint $x_j' - x_j'' = 0$, the coefficient of x_j' in the relaxed problem is $c_j + \mu_j$, and the coefficient of x_j'' is $-\mu_j$. As μ_j ranges over the real numbers, we obtain all possible ways of splitting the cost c_j between c_j' and c_j''.

The operator scheduling problem we have considered permits us to find an optimal schedule of drivers for a single bus line. If we wish to schedule several bus lines simultaneously, the right-hand-side coefficients in the constraints (16.6) will be arbitrary positive integers, indicating the number of required operators for each time period during the day. In this instance, the variable splitting device still permits us to use Lagrangian relaxation and network optimization to solve the problem. In this instance, the Lagrangian subproblems will be minimum cost flow problems rather than shortest path problems.

As this application shows, embedded network flow structure is not always so apparent and, consequently, the use of Lagrangian relaxation often requires considerable ingenuity in model formulation. Indeed, the application of Lagrangian relaxation typically requires considerable skill in modeling. Moreover, as several of our examples have shown, we often can formulate network optimization problems in several different ways, and by doing so we might be able to recognize and exploit different network substructures. The models we have proposed for the traveling salesman problem, both in the discussion of this problem and in the discussion of the vehicle routing problem, illustrate these possibilities. As a result, the design and implementation of Lagrangian relaxation algorithms often require careful choices concerning the "best" models to use and the "best" constraints to relax. The literature that we cite in the reference notes gives some guidance concerning these issues; successful prior applications, such as those that we have discussed in this section and in the exercises at the end of this chapter, provide additional guides.

Application 16.6 Degree-Constrained Minimum Spanning Trees

Suppose that we wish to find a minimum spanning tree of a network, but with the added provision that the tree contain exactly k arcs incident to a given root node, say node 1 (in some settings, the degree of the root node should be at most k). This

degree-constrained minimum spanning tree problem arises in several applications. For example, in computer networking, the root node might be a central processor with a fixed number of ports and the other nodes might be terminals that we need to connect to the processor. In the communication literature, this problem has become known as the *teleprocessing design problem* or as the *multidrop terminal layout problem*. The vehicle routing problem, described in Application 16.3, provides another application setting. If we are routing k vehicles and we delete the last arc from every route, every solution is a spanning tree with k arcs incident to the depot (the tree has additional structure: each subtree off the root is a single path). Therefore, the degree constrained minimum spanning tree problem is a relaxation of the vehicle routing problem. Note that this relaxation is stronger than the minimum spanning tree relaxation that we discussed in Application 16.3.

We might formulate the degree-constrained minimum spanning tree problem as follows.

$$\text{Minimize} \quad cx$$

subject to

$$\sum_{j=2}^{n} x_{1j} = k,$$

$$x \in X.$$

In this formulation, $x = (x_{ij})$ is a vector of decision variables and each x_{ij} is a zero–one variable indicating whether ($x_{ij} = 1$) or not ($x_{ij} = 0$), arc (i, j) belongs to the spanning tree. The number c_{ij} denotes the fixed cost of installing arc (i, j) and the set X denotes the set of incidence vectors of spanning trees. The additional constraint states that the degree of node 1 must be k. Let $C = \max\{c_{ij} : (i, j) \in A\}$.

To solve this problem, we might use Lagrangian relaxation. If we associate a Lagrange multiplier μ with the degree constraint and relax it, the objective function of the Lagrangian subproblem becomes $cx + \mu \sum_{j=2}^{n} x_{1j} - \mu k$ and the remaining (implicit) constraint, $x \in X$, states that the vector x defines a spanning tree. Note that if we ignore the last term, μk, which is a constant for any fixed value of μ, this problem is a parametric minimum spanning tree problem: for each j, the cost of arc $(1, j)$ is $c_{1j} + \mu$, and whenever $i \neq 1$ and $j \neq 1$, the cost of arc (i, j) is c_{ij}. We will use this observation to solve the degree constrained problem. That is, rather than using subgradient optimization, we will use a combinatorial algorithm to solve the Lagrangian multiplier problem.

We first solve the minimum spanning tree problem for $\mu = 0$. If the degree of node 1 in the optimal tree equals k, this tree is optimal for the degree-constrained minimum spanning tree problem. So suppose that the degree of node 1 is different than k. We first consider the case when the degree of node 1 is strictly less than k. Notice that since μ affects the lengths of only those arcs incident to node 1, changing the value μ affects the ranking of these arcs relative to the arcs not incident to node 1. Consequently, as we decrease the value of μ, the arcs incident to node 1 become more attractive relative to the other arcs, so we would insert these arcs into the spanning tree in place of the other arcs. The algorithm uses this observation: It starts with a minimum spanning tree T^1 for $\mu = 0$ and by decreasing the value of μ, it

generates a sequence of spanning trees $T^1, \ldots, T^{q-1}$, terminating with a minimum spanning tree T^q for $\mu = -C - 1$. Each tree $T^1, \ldots, T^{q-1}$ is a minimum spanning tree for some value of μ. The algorithm creates T^j from T^{j-1} by adding one arc $(1, i)$ to T^{j-1} and deleting one arc (p, q) with $p \neq 1$ and $q \neq 1$ from T^{j-1}. That is, at each step it increases the degree of node 1 by one. Finally, T^q includes all the arcs incident to node 1. (For a discussion of parametric minimum spanning trees, see Exercises 13.35 and 13.36.)

Let T^k denote the tree containing exactly k arcs incident to node 1 and let μ^k denote the value of μ for which T^k is a minimum spanning tree for the parametric problem. Further, let x^k denote the incidence vector associated with the spanning tree T^k. By definition, x^k solves the Lagrangian multiplier problem $L(\mu) = cx + \mu \sum_{j=2}^{n} x_{ij} - \mu k$, $x \in X$, for $\mu = \mu^k$ because μk is a constant. Now notice that $L(\mu^k) = cx^k + \mu^k \sum_{j=2}^{n} x_{1j} - \mu^k k = cx^k + \mu^k k - \mu^k k = cx^k$, which implies that for $\mu = \mu^k$, the optimal objective function value of the Lagrangian subproblem equals the value of a feasible solution x^k of the degree-constrained minimum spanning tree problem. Property 16.3 shows that x^k is an optimal solution of the degree-constrained minimum spanning tree problem.

When the optimal tree for $\mu = 0$ contains more than k arcs, we parametrically increase the value of μ until $\mu = C + 1$. As we increase the value of μ, the arcs incident on node 1 become less attractive, and they leave the optimal tree one by one. Eventually, node 1 will have degree exactly equal to k, and the tree at this point will be a minimum degree-constrained spanning tree.

Note that this application of Lagrangian relaxation is different than the others that we have considered in this chapter. In this case we have used Lagrangian relaxation to define a parametric problem that is related to the constrained model we are considering. We have then used a combinatorial algorithm rather than a general-purpose Lagrangian relaxation algorithm to solve the parametric problem. In this case the Lagrangian relaxation has proven to be valuable not only in formulating the parametric problem, but also in validating that the solution generated by the parametric problem is optimal for the constrained model.

Application 16.7 Multi-item Production Planning

In production planning we would like to find the best use of scarce resources (people, machinery, space) in order to meet customer demand at the least possible cost. As we show in Chapter 19, the research community has developed a number of different models for addressing various planning issues in this application domain. Some of these models are shortest path problems and some are minimum cost flow problems; still others are multicommodity flow problems or more general models with embedded network flow structure. In this section, to show how we might use Lagrangian relaxation to solve more general models, we consider two applications of production planning: multi-item production planning and production planning with changeover costs.

Suppose that we are producing K items over a planning horizon containing T periods (e.g., production shifts). Suppose, further, that we produce the items on the same machine and that we can produce at most one item in each period. We would

like to find the least cost production plan that will satisfy a demand d_{kt} for every item k in each period t.

Let x_{kt} denote the amount of item k that we produce in period t and let I_{kt} denote the amount of inventory of item k that we carry from period t to period $t + 1$. Let z_{kt} be a zero–one variable indicating whether or not we produce item k in period t. With this notation, we can model the multi-item production planning problem as follows:

$$\text{Minimize} \quad \sum_{k=1}^{K} \sum_{t=1}^{T} c_{kt}x_{kt} + \sum_{k=1}^{K} \sum_{t=1}^{T} h_{kt}I_{kt} + \sum_{k=1}^{K} \sum_{t=1}^{T} F_{kt}z_{kt} \tag{16.9a}$$

subject to

$$\sum_{k=1}^{K} z_{kt} \leq 1 \qquad \text{for } t = 1, 2, \ldots, T, \tag{16.9b}$$

$$x_{kt} + I_{k,t-1} - I_{kt} = d_{kt} \qquad \text{for } k = 1, 2, \ldots, K \text{ and } t = 1, 2, \ldots, T, \tag{16.9c}$$

$$x_{kt} \leq P_{kt}z_{kt} \qquad \text{for } k = 1, 2, \ldots, K \text{ and } t = 1, 2, \ldots, T, \tag{16.9d}$$

$$x_{kt} \geq 0, \quad I_{kt} \geq 0 \qquad \text{for } k = 1, 2, \ldots, K \text{ and } t = 1, 2, \ldots, T, \tag{16.9e}$$

$$z_{kt} = 0 \text{ or } 1 \qquad \text{for } k = 1, 2, \ldots, K \text{ and } t = 1, 2, \ldots, T. \tag{16.9f}$$

In this model c_{kt} is the per unit production cost and h_{kt} is the per unit inventory carrying cost for item k in period t. F_{kt} is a fixed cost that we incur if we produce item k in period t and P_{kt} is the production capacity for item k in period t. The constraint (16.9a) ensures that we produce at most one item in each period. Constraint (16.9c) states that we allocate the amount we have on hand of item k in period t (i.e., the production plus incoming inventory of that item) either to demand in that period or to inventory at the end of the period. The "forcing" constraint (16.9d) ensures that the quantity x_{kt} of item k produced in period t is zero if we do not select that item for production in that period, that is, if $z_{kt} = 0$; this constraint also ensures that the production of item k in period t never exceeds the production capacity of that item.

Note that constraints in (16.9b) are the only constraints in this model that link various items. Therefore, these constraints would be attractive candidates to relax via Lagrange multipliers λ_t. Doing so creates the following objective function

$$\text{Minimize} \quad \sum_{k=1}^{K} \sum_{t=1}^{T} c_{kt}x_{kt} + \sum_{k=1}^{K} \sum_{t=1}^{T} h_{kt}I_{kt} + \sum_{k=1}^{K} \sum_{t=1}^{T} [F_{kt} + \lambda_t]z_{kt} - \sum_{t=1}^{T} \lambda_t.$$

For a fixed value of the Lagrange multipliers, the last term is a constant, so the problem separates into a single-item production planning problem for each item k; the production and inventory carrying costs in the relaxation are the same as those in the original model, and in each period the fixed cost for each item in the relaxation is λ_t units more than in the original model.

The subproblems assume different forms, depending on the nature of the production capacities. In Chapter 19 we show that whenever each single-item subproblem is uncapacitated (i.e., P_{kt} is as large as the sum of the demands d_{kt} in periods $t + 1, t + 2, \ldots, T$), we can solve each single-item subproblem as a shortest path problem. If we impose production capacities, the subproblems are NP-complete. In

these instances, since the number of time periods is often very small, we might use a dynamic programming approach for solving the subproblems.

To conclude this discussion, we might note that we can enrich this basic multi-item production planning model in a variety of ways. For example, as shown in Chapter 19, we can model multiple stages of production or the backlogging of demand. As another example, we can model situations in which we incur a startup cost whenever we initiate the production of a new item. To model this situation, we let y_{kt} be a zero–one variable, indicating whether or not the production system switches from not producing item k in period $t - 1$ to producing the item in period t. We then add the following constraints to the basic model (16.9):

$$z_{kt} - z_{k,t-1} \leq y_{kt} \qquad \text{for } k = 1, 2, \ldots, K \text{ and } t = 1, 2, \ldots, T,$$

and for each "turn on" variable y_{kt}, we add a cost term $\alpha_{kt} y_{kt}$ to the objective function (α_{kt} is the cost for turning on the machine to produce item k in period t). By relaxing these constraints as well as the item choice constraints (16.9b), we again obtain separate production planning problems for each item. Or, by relaxing only the item choice constraints, we obtain a single-item production planning problem in which we incur three types of production costs: (1) a cost for turning the machine on, (2) a cost for setting up the machine in any period to produce any amount of the item, and (3) a per unit production cost.

This startup cost problem is important in many practical production settings. Moreover, this model is illustrative of the enhancements that we can make to the basic production planning problem and once again demonstrates the algorithmic flexibility of Lagrangian relaxation.

16.6 SUMMARY

Lagrangian relaxation is a flexible solution strategy that permits modelers to exploit the underlying structure in any optimization problem by relaxing (i.e., removing) complicating constraints. This approach permits us to "pull apart" models by removing constraints and instead place them in the objective function with associated Lagrange multipliers. In this chapter we have developed the core theory of Lagrangian relaxation, described popular solution approaches, and examined several application contexts in which Lagrangian relaxation effectively exploits network substructure.

The starting point for the application and theory of Lagrangian relaxation (as applied to a model specified as a minimization problem) is a key bounding principle stating that for any value of the Lagrange multiplier, the optimal value of the relaxed problem, called the Lagrangian subproblem, is always a lower bound on the objective function value of the problem. To obtain the best lower bound, we need to choose the Lagrangian multiplier so that the optimal value of the Lagrangian subproblem is as large as possible. We call this problem the Lagrangian multiplier problem. We can solve the Lagrangian multiplier problem in a variety of ways. The subgradient optimization technique is possibly the most popular technique for solving the Lagrangian multiplier problem and we have described this technique in some detail. The subgradient optimization technique solves a sequence of Lagrangian subproblems.

Usually, we choose the constraints to relax so that the Lagrangian subproblem is much easier to solve than the original problem. Consequently, when applying Lagrangian relaxation, we solve many "simple" problems instead of one single "complicated" problem. Frequently, the complicating constraints that we relax are the only constraints that couple otherwise independent subsystems (e.g., shortest path problems); in these instances, Lagrangian relaxation permits us to decompose a problem into smaller, more tractable subproblems. For this reason, the research community often refers to Lagrangian relaxation as a decomposition technique.

In discussing the theory of Lagrangian relaxation, we showed how to formulate the Lagrangian multiplier problem as an associated linear program with a large number of constraints; we also showed how to interpret the Lagrangian multiplier problem as a convexification of the original optimization model. That is, instead of restricting our choices to a discrete set of possible alternatives (e.g., spanning tree solutions), the multiplier problem produces the same objective function value that we would obtain if we solved the original problem, but permitted the use of convex combinations of the alternatives. We also showed that when applied to integer programs, the Lagrangian relaxation always gives at least as large a lower bound as does the linear programming relaxation of the problem. Finally, we showed that whenever the Lagrangian subproblem satisfies the integrality property (so it has an integer solution for all values of the Lagrange multiplier), solving the Lagrange multiplier problem is equivalent to solving the linear programming relaxation of the original optimization model. In these instances, even though the Lagrangian approach provides the same lower bound as the linear programming relaxation, it does have the ability to solve network (or other) subproblems quickly, which is often greatly preferred to solving the original problem by general-purpose linear programming codes.

Our discussion of applications has introduced several important network optimization models: networks with side constraints, the traveling salesman problem, vehicle routing, network design, personnel scheduling, degree-constrained minimum spanning trees, and production planning. As we have seen, these optimization models have applications in such diverse settings as machine scheduling, communication system design, delivery of consumer goods, telephone coin box collection, telephone operator scheduling, logistics, and production. Consequently, our discussion has illustrated the broad applicability of Lagrangian relaxation across many practical problem contexts. It has also illustrated the versatility of Lagrangian relaxation and its ability to exploit the core network substructures—shortest paths, minimum cost flows, the assignment problem, and minimum spanning tree problems—that we have studied in previous chapters. Our discussion of applications has also highlighted several other points:

1. *Need for creative modeling.* Formulating Lagrangian relaxations can require considerable ingenuity in modeling (as in the variable splitting device that we used to study the two-duty operator scheduling problem).
2. *Flexibility of Lagrangian relaxation.* In many models, such as the vehicle routing problem, we can obtain a variety of different Lagrangian subproblems by relaxing different constraints. This variety of potential subproblems permits us to develop different algorithms for solving the same problem.

3. *Use of Lagrangian relaxation as a conceptual as well as algorithmic tool.* On some occasions, as in our discussion of the degree-constrained minimum spanning tree problem, we can use the bounding information provided by Lagrangian relaxation as a stand-alone tool that is unrelated to any iterative method for solving the Lagrangian multiplier problem. For example, we can use the bounds to analyze the solutions generated by combinatorial or heuristic algorithms for solving a problem.

REFERENCE NOTES

The Lagrange multiplier technique of nonlinear optimization dates to the eighteenth century and was suggested by the famous mathematician Lagrange, for whom the technique is named. The use of this technique in integer programming and discrete optimization is much more recent, originating in the seminal papers by Held and Karp [1970, 1971], who studied the traveling salesman problem. Everett's [1963] development of Lagrangian multiplier methods for general mathematical programming problems was a precursor to this development. Held and Karp's application of the Lagrange multiplier method was not only an eye-opening successful application, but also set out many key ideas in applying the method to integer programming problems. Fisher [1981, 1985], Geoffrion [1974], and Shapiro [1979] provide insightful surveys of Lagrangian relaxation and its uses in integer programming. The papers by Fisher contain many citations to successful applications in a wide variety of problem settings. For a discussion of the branch-and-bound algorithm, see Winston [1991].

Most of the key results of Lagrangian relaxation (e.g., the bounding properties and optimality conditions) are special cases of more general results in mathematical programming duality theory. Rockafellar [1970] and Stoer and Witzgall [1970] provide comprehensive treatments of this subject. Magnanti, Shapiro, and Wagner [1976] establish the equivalence of the Lagrangian multiplier problem and generalized linear programming, whose development by Dantzig and Wolfe [1961] predates the formal development of Lagrangian relaxation in integer programming. The integrality property is due to Geoffrion [1974]. The subgradient method is an outgrowth of so-called relaxation methods for solving systems of linear inequalities. Bertsimas and Orlin [1991] have developed the most efficient algorithms (in the worst-case sense) for solving many classes of Lagrangian relaxation problems.

Several of the application contexts that we have discussed in Section 16.5 and in the exercises have very extensive literatures. The following books and survey articles, which contain many references to the literature, serve as good sources of information on these topics.

Traveling salesman problem: the book edited by Lawler, Lenstra, Rinnooy Kan, and Shmoys [1985]

Vehicle routing: surveys by Bodin, Golden, Assad, and Ball [1983], Laporte and Nobert [1987], and Magnanti [1981]

Network design: surveys by Magnanti and Wong [1984], Magnanti, Wolsey, and Wong [1992], and Minoux [1989]

Production planning: the survey paper by Shapiro [1992], the book by Hax and Candea [1984], and the paper by Graves [1982]

Several other of the applications discussed in this chapter are adapted from research papers from the literature. For a Lagrangian relaxation-based branch-and-bound approach to the constrained shortest path problem, see Handler and Zang [1980]. Shepardson and Marsten [1980] have used the variable splitting device and Lagrangian relaxation for solving the two-duty operator scheduling problem and applied this approach to bus operator scheduling. For an algorithmic approach to the network design problem, see Balakrishnan, Magnanti, and Wong [1989a]. Volgenant [1989] considers the degree-constrained minimum spanning tree problem.

EXERCISES

16.1. Lagrangian relaxation and inequality constraints. To develop the Lagrangian multiplier problem for an inequality constraint problem stated as $\min\{cx : \mathcal{A}x \leq b, x \in X\}$, suppose that we add nonnegative "slack" variables s to model the problem in the following equivalent equality form : $\min\{cx : \mathcal{A}x + s = b, x \in X \text{ and } s \geq 0\}$.

(a) State the Lagrangian multiplier problem for the equality formulation.

(b) Show that if some $\mu_i < 0$, then $L(\mu) = -\infty$. Further, show that if some $\mu_i > 0$, then in the optimal solution of the Lagrangian subproblem $L(\mu)$, the slack variable $s_i = 0$.

(c) Conclude from part (b) that the Lagrangian multiplier problem of the inequality constrained problem is $\max_{\mu \geq 0} L(\mu)$ with $L(\mu) = \min\{cx + \mu(\mathcal{A}x - b) : x \in X\}$.

16.2. Consider the problem

$$\text{Minimize} \quad -2x - 3y$$

subject to

$$x + 4y \leq 5,$$

$$x, y \in \{0, 1\},$$

and the corresponding relaxed problem

$$\text{Minimize} \quad -2x - 3y + (x + 4y - 5)$$

subject to

$$x, y \in \{0, 1\}.$$

Show that $x = 1, y = 0$ solves the relaxed problem, is feasible for the original problem, and yet does not solve the original problem. (Reconcile this example with Property 16.4.)

16.3. Lagrangian relaxation applied to linear programs. Suppose that we apply Lagrangian relaxation to the linear program $\mathcal{P}$ defined as $\min\{cx : \mathcal{A}x = b, x \geq 0\}$ by relaxing the equality constraints $\mathcal{A}x = b$. The Lagrangian function is $L(\mu) = \min_{x \geq 0} \{cx - \mu(\mathcal{A}x - b)\} = \min_{x \geq 0} \{(c - \mu\mathcal{A})x + \mu b\}$. (Since the constraints $\mathcal{A}x = b$ are equalities, the Lagrange multipliers μ are unconstrained in sign. For the purpose of this exercise, we have chosen a different sign convention than usual, that is, used $-\mu$ in place of μ.) Now, consider the Lagrangian multiplier problem $\max_{\mu} L(\mu)$.

(a) Suppose we choose a value of μ so that for some j, $(c - \mu\mathcal{A})_j < 0$. Show that $L(\mu) = -\infty$.

(b) Suppose we choose a value of μ so that for some j, $(c - \mu\mathcal{A})_j > 0$. Show that in the optimal solution of the Lagrangian subproblem, $x_j = 0$.

(c) Conclude from parts (a) and (b) that the Lagrangian multiplier problem is equiv-

alent to the linear programming dual of $\mathcal{P}$, that is, the problem max μb, subject to $\mu \mathcal{A} \leq c$.

16.4. Oscillation in Lagrangian relaxation. Suppose that we apply Lagrangian relaxation to the constrained shortest path example shown in Figure 16.1 with the time constraint of $T = 14$, starting with value $\mu^0 = 0$ for the Lagrange multiplier μ. Show that if we choose the step size $\theta_k = 1$ at each iteration, the subgradient algorithm $\mu^{k+1} = \mu^k + \theta_k(\mathcal{A}x^k - b)$ oscillates between the values $\mu = 0$ and $\mu = 4$ and the Lagrangian subproblem solutions alternate between the paths 1–2–4–6 and 1–3–2–5–6.

16.5. In Section 16.4 we showed that when $T = 14$, our constrained shortest path example had an optimal objective function value $z^* = 13$ while the Lagrange multiplier problem had a value $L^* = 7$. Show that L^* equals the optimal objective function value of the linear programming relaxation of the problem. Interpret the solution of the linear program as the convex hull of shortest path solutions. That is, find a set of paths whose convex combination satisfies the timing constraint and whose weighted (i.e., convex combination) cost equals L^*.

16.6. Suppose that X is a finite set and that when we solve the Lagrangian multiplier problem corresponding to the optimization problem $\min\{cx : \mathcal{A}x = b, x \in X\}$ for *any* value of c, we find that the problem has no duality gap, that is, if x^* solves the given optimization problem and μ^* is an optimal solution to the Lagrangian multiplier problem, then $cx^* = L(\mu^*)$. Show that the polyhedron $\{x : \mathcal{A}x = b \text{ and } x \in \mathcal{H}(X)\}$ has integer extreme points. (*Hint*: Use the results given in the proofs of Theorems 16.9 and 16.10.)

16.7. Lagrangian relaxation interpretation of successive shortest paths. Recall from Section 9.7 that each intermediate stage of the successive shortest path algorithm for solving the minimum cost flow problem maintains a pseudoflow x satisfying the flow bound constraints and a vector π of node potentials satisfying the conditions $c_{ij}^{\pi} = c_{ij} - \pi(i) + \pi(j) \geq 0$ for all arcs $(i, j) \in G(x)$.

(a) Show that the pseudoflow x is optimal for the problem obtained by relaxing the mass balance constraints and replacing the objective function cx with the Lagrangian function

$$\text{Minimize} \sum_{(i,j)\in A} (c_{ij} - \pi(i) + \pi(j))x_{ij}.$$

(b) Interpret the successive shortest path algorithm as a method that proceeds by adjusting the Lagrangian multipliers. At each stage the method adjusts the multipliers π so that (1) the current pseudoflow x is optimal for the Lagrangian subproblem, and (2) some alternate optimal pseudoflow x' for the Lagrangian relaxation is "less infeasible" than x. Finally, when the optimal pseudoflow becomes a flow, we obtain an optimal solution of the Lagrangian subproblem that is also feasible for the original problem; therefore, it must be an optimal solution of the original problem.

16.8. Generalized assignment problem (Ross and Soland [1975]). The generalized assignment problem is the optimization model

$$\text{Minimize} \sum_{i\in I} \sum_{j\in J} c_{ij}x_{ij} \tag{16.10a}$$

subject to

$$\sum_{j\in J} x_{ij} = 1 \qquad \text{for all } i \in I, \tag{16.10b}$$

$$\sum_{i\in I} a_{ij}x_{ij} \leq d_j \qquad \text{for all } j \in J, \tag{16.10c}$$

$$x_{ij} \geq 0 \text{ and integer} \qquad \text{for all } (i, j) \in A. \tag{16.10d}$$

In this problem we wish to assign $|I|$ "objects" to $|J|$ "boxes." The variable $x_{ij} = 1$ if we assign object i to box j and $x_{ij} = 0$ otherwise. We wish to assign each

object to exactly 1 box; if assigned to box j, object i consumes a_{ij} units of a given "resource" in that box. The total amount of resource available in the jth box is d_j. This generic model arises in a variety of problem contexts. For example, in machine scheduling, the objects are jobs, the boxes are machines; a_{ij} is the processing time for job i on machine j and d_j is the total amount of time available on machine j.

(a) Outline the steps required for solving the Lagrangian subproblem obtained by (1) relaxing the constraint (16.10b), and (2) by relaxing the constraint (16.10c).

(b) Compare the lower bounds obtained by the two relaxations suggested in part (a). Which provides the sharper lower bound? Why? (*Hint*: Use Theorems 16.9 and 16.10.)

(c) Compare the optimal objective function value of the Lagrangian multiplier problem for each relaxation suggested in part (a) with the bound obtained by the linear programming relaxation of the generalized assignment model.

16.9. Facility location (Erlenkotter [1978]). Consider the following facility location model:

$$\text{Minimize} \quad \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} + \sum_{j \in J} F_j y_j \tag{16.11a}$$

subject to

$$\sum_{j \in J} x_{ij} = 1 \qquad \text{for all } i = 1, 2, \ldots, I, \tag{16.11b}$$

$$\sum_{i \in I} d_i x_{ij} \leq K_j y_j \qquad \text{for all } j = 1, 2, \ldots, J, \tag{16.11c}$$

$$0 \leq x_{ij} \leq 1 \qquad \text{for all } i \in I \text{ and } j \in J, \tag{16.11d}$$

$$y_j = 0 \text{ or } 1 \qquad \text{for all } j \in J. \tag{16.11e}$$

In this model, I denotes a set of customers and J denotes a set of potential facility (e.g., warehouse) locations used to supply to the customers. The zero–one variable y_j indicates whether or not we choose to locate a facility at location j and x_{ij} is the fraction of the demand of customer i that we satisfy from facility j. The constant d_i is the demand of customer i. The cost coefficient c_{ij} is the cost (e.g., the transportation cost) of satisfying all of the ith customer's demand from facility j, and the cost coefficient F_j is the fixed cost of opening (e.g., leasing) a facility of size K_j at location j. The constraints (16.11b) state that we need to satisfy all of the demand for each customer, and the constraints (16.11c) state that (1) we cannot meet any of the demand of any customer if we do not locate a facility at location j (i.e., $x_{ij} = 0$ if $y_j = 0$), and (2) if we do locate a facility at location j (i.e., $y_j = 1$), the total demand met by the facility cannot exceed the facility's capacity K_j.

(a) Show how you would solve the Lagrangian subproblem obtained by relaxing the constraints (16.11b). (*Hint*: Note that the Lagrangian subproblem decomposes into a separate subproblem for each location.)

(b) Show next how you would solve the Lagrangian subproblem if we relax the constraints (16.11c). (*Hint*: Note that the Lagrangian subproblem decomposes into a separate subproblem for each customer.)

(c) Show that if $|I| = |J| = 1$, $K_1 = 10$, $d_1 = 5$, and $c_{11} = 0$, the relaxation suggested in part (a) gives a sharper lower bound than the relaxation in part (b). Next prove the general result that the relaxation in part (a) gives at least as good a bound as given by the relaxation in part (b).

16.10. Modified facility location. Suppose that in the model considered in Exercise 16.9, we impose the additional constraint that the demand for each customer should be "sole sourced"; that is, each variable x_{ij} has value zero or 1.

(a) Show how to use the solution of a single knapsack problem for each facility j to solve the Lagrangian relaxation obtained by relaxing the constraints (16.11b).

(b) Show that the bound obtained from the Lagrangian multiplier problem by relaxing

the constraints (16.11b) is always at least as strong as the bound obtained by relaxing the constraints (16.11c).

16.11 Tightening the facility location relaxation. Suppose that we add the redundant constraints $x_{ij} \leq \min\{y_j, K_j\}$ to the facility location model described in Exercise 16.9 and then we apply Lagrangian relaxation by relaxing the constraints (16.11b) or (16.11c).

(a) Show that the bound obtained from the Lagrangian multiplier problem is always as strong or stronger than the bound obtained by relaxing the corresponding constraints in the original model without the additional constraints $x_{ij} \leq \min\{y_j, K_j/d_i\}$.

(b) How would you solve the Lagrangian subproblem with the added constraints $x_{ij} \leq \min\{y_j, K_j/d_i\}$?

(c) How would your answers to parts (a) and (b) change if we considered the sole-sourcing-facility location model described in Exercise 16.10?

16.12. Local access capacity expansion (Balakrishnan, Magnanti, and Wong [1991]). The lowest level of national telephone networks are trees that connect individual customers to the rest of the national network through special nodes known as *switching centers*, which route telephone calls to their final destination. Each local access network (tree) T has its own switching center. As demand for service increases, telephone companies have two basic options for increasing the capacity of a local access network: (1) they can install more copper cables on the arcs of the networks; or (2) they can install devices, called *multiplexers* (or *concentrators*), at the nodes. The multiplexers compress calls so that they use less downstream cable capacity. We assume that once a call reaches a multiplexer, it requires negligible cable capacity to send it to the switching center. Every call must be routed through the tree T either to the switching center or to one of the multiplexers. Suppose that the existing capacity of arc (i, j) is u_{ij} and increasing the capacity by y_{ij} units incurs an arc-dependent cost $c_{ij}y_{ij}$. Let d_i denote the numbers of calls originating at node i that must be routed to the switching center or to a multiplexer. Each multiplexer has two associated costs: (1) a fixed cost F, and (2) a variable throughput cost α incurred for each unit of call compressed by that multiplexer. The optimization problem is to meet the demand for service by incurring minimum total cost.

(a) Let z_i be a zero–one variable indicating whether or not we place a multiplexer at node i. Further, let x_{ij} be a zero–one variable indicating whether or not we assign node i to the multiplexer j. In the local access network T, for any pair $[i, j]$ of nodes, we let P_{ij} denote the unique path between these two nodes, and for any arc (k, l) in T, we let Q_{kl} be the set of all node pairs $[i, j]$ from which P_{ij} contains the arc (k, l). We assume that node 1 is the switching center. Let node S denote the remaining nodes in the network. Using this notation, give an integer programming formulation of the local access network design problem.

(b) Suggest two relaxations of the formulation in part (a) that produce a relaxed problem with a structure that we have treated in this book.

16.13. Contiguous local access capacity expansion problem (Balakrishnan, Magnanti, and Wong [1991]). In most practical settings of the local access capacity expansion problems, the set of nodes assigned to the switching center or to any multiplexer must be contiguous. That is, if we assign node i to a multiplexer at node j and node k lies on the path in T from node i to node j, we must also assign node k to the multiplexer at node j. Therefore, the final configuration of the local access network will be a subdivision of the tree T into subtrees, with each subtree containing either the switching center or one multiplexer and the nodes it serves.

(a) Show that we can incorporate the contiguity condition in the formulation of Exercise 16.12 by adding the following constraints for every pair $[i, j]$ of nodes: if the path P_{ij} contains node k, then $x_{kj} \geq x_{ij}$.

(b) Consider the integer programming formulation of the contiguous local access capacity expansion problem from part (a). Suppose that we relax the capacity constraints imposed on the arcs. Show that we can solve the Lagrangian subproblem in polynomial time using a dynamic programming technique.

16.14. Design of telecommunication networks (Leung, Magnanti, and Singhal [1990] and Magnanti, Mirchandani, and Vachani [1991]). In designing telecommunications networks, we would like to install sufficient capacity to carry required traffic (telephone calls, data transmissions) simultaneously between various source–sink locations. Suppose that (s^k, t^k) for $1 \le k \le K$ denote K pairs of source–sink locations, and r^k denotes the number of messages sent from the source s^k to the sink t^k. We can install either of two different types of facilities on each link of the transmission network, so-called T0 lines and T1 lines. Each T0 line can carry 1 unit of message and each T1 line can carry 24 units of messages; installing a T0 line on arc (i, j) incurs a cost of a_{ij} and installing a T1 line on arc (i, j) incurs a cost of b_{ij}. Once we have installed the lines, we incur no additional costs in sending flow on them. This problem arises in practice because companies with large telecommunication requirements might be able to lease lines more cost-effectively than paying public tariffs. The same type of problem arises in trucking of freight; in this setting, the facilities to be "installed" on any arc are the trucks of a particular type (e.g., 36-foot trailers or 48-foot trailers) to be dispatched on that arc.

(a) Show how to formulate this telecommunication network design problem with two types of constraints: (1) a set of network flow constraints modeling the required flow between every pair of source–sink locations, and (2) capacity constraints restricting the total flow on each arc to be no more than the capacity that we install on that arc. (*Hint*: Use the following integer decision variables: (1) y_{ij} : the number of T0 lines for arc (i, j), (2) z_{ij} : the number of T1 lines for arc (i, j), and (3) the number of messages x_{ij}^k sent from the source s^k to the sink t^k that pass through the arc (i, j).)

(b) How would you solve the linear programming relaxation of this model? (*Hint*: Consider two cases: when $24\ a_{ij} < b_{ij}$ and when $24\ a_{ij} \ge b_{ij}$.)

(c) Show how to solve the Lagrangian subproblem obtained by relaxing constraints of type 1 in the model formulation. (*Hint*: Consider the two cases as in part (b).)

(d) Show how to solve the Lagrangian subproblem obtained by relaxing constraints of type 2 in the model formulation.

16.15. Steiner tree problem. The Steiner tree problem is an $\mathcal{NP}$-hard variant of the minimum spanning tree problem. In this problem we are given a subset $S \subseteq N$ of nodes, called *customer nodes*, and we wish to determine a minimum cost tree (not necessarily a spanning tree) that must contain all the nodes in S and, optionally, some nodes in $N - S$. This problem arises in many application settings, such as the design of rural road networks, pipeline networks, or communication networks. Formulate this problem as a special case of the network design problem discussed in Application 16.4 and show how to apply Lagrangian relaxation to the resulting formulation. (*Hint*: Designate any customer node as a source node and send 1 unit of flow to every other customer node.)

16.16. In this exercise we show how to formulate a directed traveling salesman problem as a network design problem. Consider a network design problem with the following data: (1) unit commodity flow requirements between every pair of nodes, (2) the cost of flow on every arc is zero, and (3) the fixed cost of the arc (i, j) is $c_{ij} + M$ for some sufficiently large number M. Show that the optimal network will be an optimal traveling salesman tour with c_{ij} as arc lengths. (*Hint*: The optimal network must be strongly connected and must contain the fewest possible number of arcs.)

16.17. Uncapacitated undirected network design problem. In the formulation of the directed uncapacitated network design problem in Application 16.4, the zero–one vector y_{ij} indicated whether we would include the directed arc (i, j) in the underlying network. Suppose, instead, that the arcs are undirected, so if we introduce arc (i, j) in the network, we can send flow in either direction on the arc.

(a) How would you formulate this problem and apply Lagrangian relaxation to obtain lower bounds?

(b) Show that in the uncapacitated undirected network design problem, if all flow costs c_{ij}^k are zero, all the fixed costs f_{ij} are nonnegative, and the problem has a commodity for each pair of nodes, the problem reduces to the minimum spanning tree problem.

16.18. (a) Show that if the uncapacitated network design problem has a single commodity (i.e., $K = 1$), we can solve the problem by solving a single shortest path problem.

(b) Show how to formulate the production planning problems that we described in Application 16.7 as capacitated or uncapacitated network design problems.

16.19. (a) Suppose that we relax the mass balance constraint $\mathcal{N}x = b$ in the formulation (16.3) of the traveling salesman problem described in Application 16.2. Show how to solve the Lagrangian subproblem as an assignment problem. (*Hint*: Show that some optimal solution of the Lagrangian subproblem satisfies the conditions $x_{ij} = 0$ or $x_{ij} = (n - 1)y_{ij}$ for each arc $(i, j) \in A$. Use this fact to eliminate the variables x_{ij} from the subproblem.)

(b) Suppose that we relax the assignment constraints (16.3b) and (16.3c) in the formulation (16.3) of the traveling salesman problem. Show how to solve the Lagrangian subproblem as an uncapacitated network design problem.

(c) What is the relationship between the optimal solutions of the Lagrangian multiplier problems obtained by the relaxations considered in parts (a) and (b), the relaxation described in the text (obtained by relaxing the forcing constraints $x_{ij} \leq (n - 1)y_{ij}$), and the optimal objective function value of the linear programming relaxation of the problem?

16.20. Assignment-based formulation of the traveling salesman problem

(a) Consider the integer program (16.3) with the constraints (16.3d) and (16.3e) replaced by the subtour breaking constraints (16.4h). Show that the resulting model is an integer programming formulation of the traveling salesman problem.

(b) Show that the solution of the Lagrangian subproblem formed by relaxing the subtour breaking constraints will be a set of directed cycles satisfying the property that each node is contained in exactly one cycle. Describe a heuristic method for modifying the Lagrangian subproblem solution so that it becomes a feasible traveling salesman tour.

16.21. Undirected traveling salesman problem. In the undirected (or symmetric) traveling salesman problem, we can traverse any arc (i, j) in either direction at the same cost c_{ij}. Let y_{ij} indicate whether or not we include arc (i, j) in a feasible tour.

(a) Give a formulation of this problem as an integer program containing three sets of constraints: (1) degree 2 constraints, indicating that each node should have degree at most 2 in any feasible tour; (2) subtour breaking constraints on the nodes $2, 3, \ldots, n$; and (3) a cardinality constraint indicating that the tour contains exactly n arcs. (*Hint*: Modify the assignment based formulation of the directed traveling salesman problem described in Exercise 16.20.)

(b) Show how to apply Lagrangian relaxation in two ways: (1) by relaxing the degree 2 constraints; and (2) by relaxing the subtour breaking constraints and the cardinality constraint. In case 1 show how to solve the Lagrangian subproblem as a 1-tree (see Exercise 13.38). In case 2 show how to solve the Lagrangian subproblem as a matching problem. (*Hint*: In case 2, first show that any network with n arcs and degree at most 2 on each node, must have a degree of exactly 2 at each node.)

16.22. Multicommodity flow-based formulation of the traveling salesman problem. In Application 16.2 we examined a single-commodity flow-based formulation of the traveling salesman problem with $n - 1$ units available at a source node (which we arbitrarily took to be node 1) and 1 unit of demand required at each other node. Suppose that, instead, we formulated a multicommodity flow model with $2(n - 1)$ commodities, with two commodities k defined for each node $k \neq 1$, an "outgoing" commodity and an "incoming" commodity. The incoming commodity for node k has 1 unit of supply at node 1 and 1 unit of demand at node k, and the outgoing commodity for node k has

1 unit of supply at node k and 1 unit of demand at node 1 (i.e., we wish to send 1 unit from node 1 to node k and 1 unit from node k to node 1). We can state this formulation of the traveling salesman problem as the following integer program:

$$\text{Minimize} \quad \sum_{(i,j)\in A} c_{ij}y_{ij},$$

subject to

$$\sum_{1\le j\le n} y_{ij} = 1 \qquad \text{for all } i = 1, 2, \ldots, n,$$

$$\sum_{1\le i\le n} y_{ij} = 1 \qquad \text{for all } j = 1, 2, \ldots, n,$$

$$Nx^k = b^k \qquad \text{for all } k = 2, \ldots, n,$$

$$Nz^k = d^k \qquad \text{for all } k = 2, \ldots, n,$$

$$x_{ij}^k \le y_{ij} \text{ and } z_{ij}^k \le y_{ij} \qquad \text{for all } (i, j) \text{ and all } k,$$

$$y_{ij} \text{ and } x_{ij}^k = 0 \text{ or } 1 \qquad \text{for all } (i, j) \text{ and all } k.$$

Note that the supply/demand vectors b^k and d^k in this formulation have a special form: $d^k = -b^k$ and b_i^k is 1 if $i = 1$, is -1 if $i = k$, and is 0 if $i \neq 1$ and $i \neq k$.

(a) Suppose that we apply Lagrangian relaxation to the multicommodity flow-based model by relaxing the forcing constraints [i.e., the constraints $x_{ij}^k \le y_{ij}$ and $z_{ij}^k \le y_{ij}$, for all (i, j) and all k]. How would you solve the Lagrangian subproblem?

(b) Show that the lower bound L^* determined by the Lagrangian multiplier problem for the Lagrangian relaxation in part (a) is always as strong or stronger than the lower bound determined by relaxing the forcing constraints in the single-commodity flow-based formulation. (*Hint*: Compare the set of feasible solutions of both problems.)

(c) What is the relationship between the optimal objective function values of the linear programming relaxations of the single and multicommodity flow-based formulations? (*Hint*: Same as that in part (b).)

16.23. Alternate formulations of the traveling salesman problem (Wong [1980]). In this chapter we have considered three different formulations of the traveling salesman problem: (1) a single-commodity flow-based formulation in Application 16.2, (2) an assignment-based formulation discussed in Exercise 16.20, and (3) a multicommodity flow-based formulation in Exercise 16.22, where we showed that from the perspective of linear programming or Lagrangian relaxations, the multicommodity flow-based formulation is stronger than the single commodity flow-based formulation.

(a) Show that we can replace the subtour breaking constraints in the assignment-based formulation, or in its linear programming relaxation, by the constraints $\sum_{i\in S} \sum_{j\in N-S} y_{ij} \ge 1$ for all sets S of nodes satisfying the cardinality condition $1 \le |S| \le n - 1$, and in both cases obtain an equivalent model (i.e., one with the same feasible solutions).

(b) Using the max-flow min-cut theorem and part (a), show that the linear programming relaxation of the assignment-based formulation of the traveling salesman problem and the linear programming relaxation of the multicommodity flow-based formulation are equivalent in the sense that y is feasible in the linear programming relaxation of the assignment-based formulations if and only if for some flow vector x, (x, y) is feasible in the multicommodity flow-based formulation. (Note that the number of subtour breaking constraints in the assignment-based formulation is exponential in n. The number of constraints in the multicommodity flow-based formulation is polynomial in n, so this formulation is a so-called *compact* formulation.)

16.24. Consider the undirected traveling salesman problem shown in Figure 16.11.

(a) What is the optimal tour length for this problem?

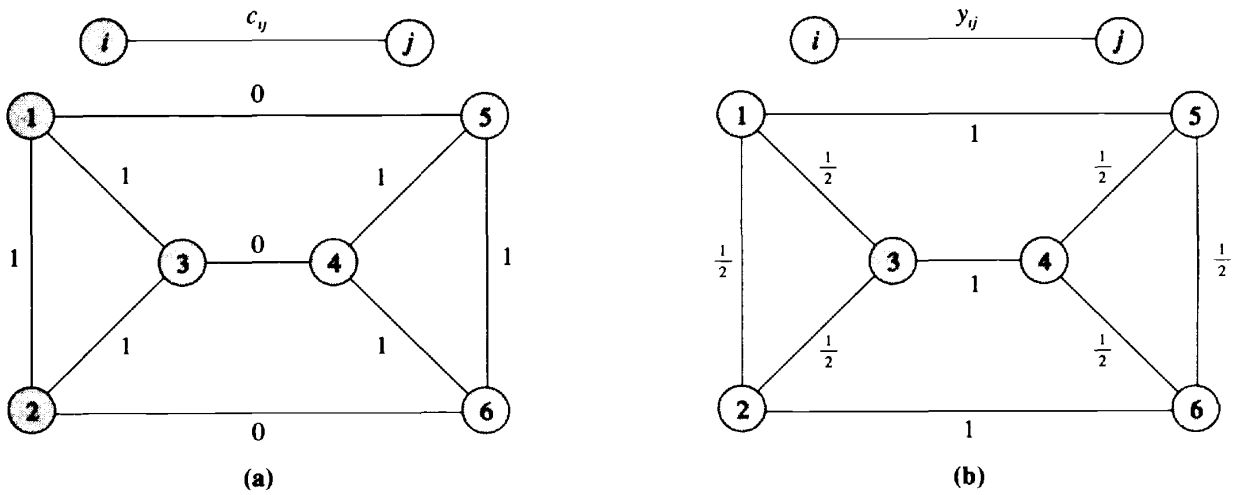

Figure 16.11 Traveling salesman problem: (a) network data; (b) solution to the linear programming relaxation.

(b) Show that the arc weights shown in Figure 16.11(b) solve the linear programming relaxation of the formulation developed in Exercise 16.21. Interpret this solution as the convex hull of 1-tree solutions to the Lagrangian subproblem that we obtain by relaxing the degree two constraints in this formulation. That is, show how to represent this solution as a convex combination of 1-tree solutions.

(c) Show the network corresponding to the equivalent directed traveling salesman problem and specify the optimal solution to the linear programming relaxation of the assignment-based formulation and both the single and multicommodity flow-based formulations.

(d) Interpret the solution to each linear programming relaxation as the convex hull of solutions to Lagrangian subproblems.

16.25. *K*-traveling salesman problem. Suppose that we wish to find a set of K arc-disjoint directed cycles in a directed graph satisfying the property that node 1 is contained in exactly K cycles and every other node is contained in exactly one cycle. In this model each arc has an associated cost c_{ij} and we wish to find a feasible solution with the smallest possible sum of arc costs. We refer to this problem as the K-traveling salesman problem since it corresponds to a situation in which K salesmen, all domiciled at the same node 1, need to visit all the other nodes of a graph.

(a) Formulate this problem as an optimization model and show how to apply Lagrangian relaxation to the formulation. (*Hint*: Modify the single-commodity flow-based formulation given in Application 16.2 or the assignment-based formulation given in Exercise 16.20.)

(b) By forming K copies of node 1 and assigning a large cost with all of the arcs joining the copies of node 1, show how to formulate the problem as an equivalent (single) traveling salesman problem.

16.26. Consider the K-traveling salesman problem described in Exercise 16.25. Show how to formulate this problem as a special case of the vehicle routing problem described in Application 16.3. The resulting formulation will be considerably simpler than the general vehicle routing problem. (*Hint*: First show that we can eliminate the constraints (16.4g). Next show how to use the constraints (16.4b) to eliminate the variables x_{ij}^k.)

16.27. Vehicle routing with nonhomogeneous fleets and with time restrictions. This exercise studies a generalization of the vehicle routing problem discussed in Application 16.3. Show how to formulate a vehicle routing problem with each of the following problem ingredients: (1) each vehicle k in a fleet of K vehicles can have different capacity u^k,

or (2) each vehicle must make its deliveries within T hours, given that it takes t_{ij} hours to traverse any arc (i, j).

16.28. Suppose that we add the redundant constraint $\sum_{(i,j)\in A} y_{ij} = n + K$ to the formulation (16.4) of the vehicle routing problem. Consider the additional set of constraints $\sum_{j\in S} y_{1j} + \sum_{i\in S} \sum_{j\in S} y_{ij} \leq |S|$ for all subsets S of $\{2, 3, \ldots, n\}$.

(a) Are these constraints valid?

(b) Are these constraints implied by the other constraints in the integer programming formulation of the problem? Are they implied by the other constraints in the linear programming relaxation of the problem?

(c) Suppose that we add the additional constraints to the formulation of the vehicle routing problem. Show that by relaxing the capacity constraints (16.4g) and the assignment constraints (16.2c) and (16.2d) for nodes 2, 3, . . . , n, the resulting Lagrangian subproblem decomposes into two subproblems: (i) a degree-constrained minimum spanning tree problem with degree K imposed on node 1, and (ii) a problem of choosing the K cheapest (with respect to the Lagrangian subproblem coefficients) arcs of the form y_{j1}. (*Hint*: First eliminate the variables x_{ij}^k and then note that the Lagrangian subproblem decomposes into two subproblems, one containing the variables y_{j1} and one containing all the other variables.)

16.29. Solve the degree-constrained minimum spanning tree problem shown in Figure 16.12 assuming that the degree of node 1 must be 8. Solve it if the degree of node 1 must be 5.

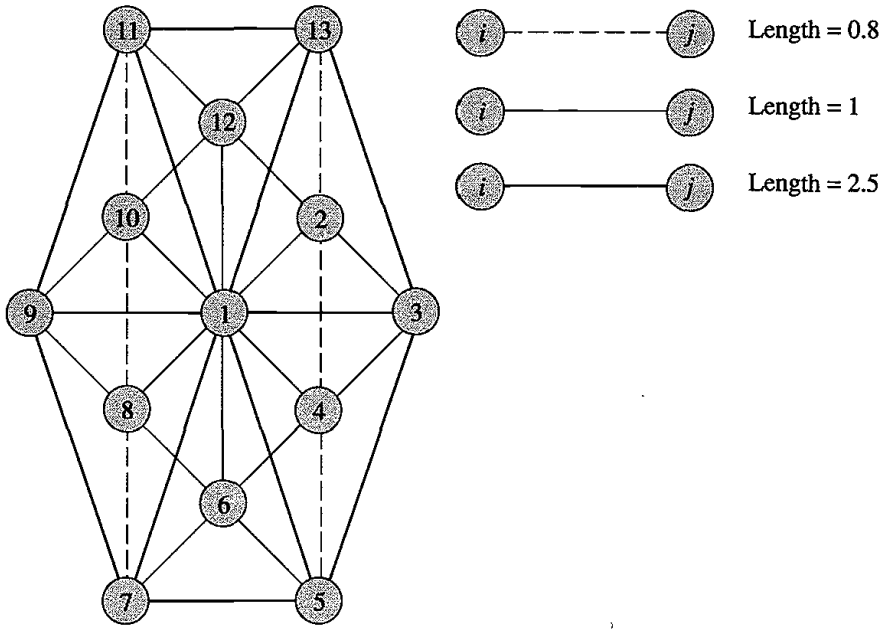

Figure 16.12 Constrained minimum spanning tree problem.

16.30. Suppose that X is a finite set and that when we solve the Lagrangian multiplier problem corresponding to the optimization problem $\min\{cx : \mathcal{A}x = b, x \in X\}$ for *any* value of c, we find that the problem has no relaxation gap; that is, if x^* solves the given optimization problem and μ^* is an optimal solution to the Lagrangian multiplier problem, then $cx^* = L(\mu^*)$. Show that the polyhedron $\{x \in \mathcal{H}(X) : \mathcal{A}x = b\}$ has integer extreme points. (*Hint*: Use the equivalence between convexification and Lagrangian relaxation (Theorem 16.10) and the fact that every extreme point to a polyhedron $\mathcal{P}$ is the *unique* optimal solution to the linear program $\min\{cx : x \in \mathcal{P}\}$ for *some* choice of the objective coefficients c.)

16.31. Let X denote the set of incidence vectors of spanning trees of a given network.

(a) Using Exercise 16.30 and the results in Section 16.4, show that for any value of k, the polyhedron $\{x : x \in \mathcal{H}(X) \text{ and } \sum_{j \neq 1} x_{ij} = k\}$ has integer extreme points. Note that if we view the set of solutions to $x \in \mathcal{H}(X)$ and $\sum_{j \neq 1} x_{1j} = k$ as we vary k as "parallel slices" through the polyhedron $x \in \mathcal{H}(X)$, this result says that extreme points of every slice are integer valued.

(b) For any subset S of nodes, let $A(S) = \{(i, j) \in A : i \in S \text{ and } j \in S\}$. Using the result of part (a) and the development in Section 13.8, show that for any value of k, the following polyhedron has integer extreme points:

$$\sum_{j \neq 1} x_{1j} = k,$$

$$\sum_{(i,j) \in A} x_{ij} = n - 1,$$

$$\sum_{(i,j) \in A(S)} x_{ij} \leq |S| - 1 \qquad \text{for any set } S \text{ of nodes,}$$

$$x_{ij} \geq 0.$$

(*Hint*: In Section 13.8 we showed that without the cardinality constraint $\sum_{j \neq 1} x_{1j} = k$, the extreme points in the polyhedron defined by the remaining constraints are incident vectors of spanning tree solutions (and so are integer valued).)

16.32. Suppose that we wish to find a minimum spanning tree of an undirected graph G satisfying the additional conditions that the degree of node 1 is k and the degree of node n is l. Suggest a Lagrangian relaxation bounding procedure for this problem. (*Hint*: Consider relaxing just one of the two degree constraints.)

16.33. Capacitated minimum spanning tree problem (Gavish [1985]). In some applications of the minimum spanning tree problem, we want to construct a capacitated tree T rooted at a specially designated node, say node 1. In this problem we wish to identify a minimum cost spanning tree subject to the additional condition that no subtree of T formed by eliminating all the arcs incident to node 1 contains more than a prescribed number u of nodes. This model arises, for example, in computer networking when node 1 is a central processor and for reasons of reliability we wish to limit the number of nodes (terminals) attached to this node through any of its ports (incident arcs). Let y_{ij} be a zero–one variable, indicating whether or not we include arc (i, j) in the optimal capacitated tree.

(a) Explain how the capacitated minimum spanning tree problem differs from the degree-constrained minimum spanning tree problem.

(b) By introducing additional constraints in the integer programming formulation (13.2) of the minimum spanning tree problem, obtain an integer programming formulation of the capacitated minimum spanning tree problem.

(c) Suggest a Lagrangian-based method for obtaining a lower bound on the optimal solution by solving a sequence of minimum spanning tree problems.

16.34. Identical customer vehicle routing problem. In the identical customer vehicle routing problem, each customer has the same demand. Formulate this problem as a capacitated minimum spanning tree problem with additional constraints. Show how to obtain bounds on the objective values by applying Lagrangian relaxation to this problem using the capacitated minimum spanning tree problem as a subproblem.

16.35. Note that every solution to a vehicle routing problem is a degree constrained minimum spanning tree (with degree K for node 1) together with K additional arcs incident to node 1 as well as another set of constraints modeling vehicle capacities. Use this observation to give a formulation of the vehicle routing problem and an associated Lagrangian relaxation that contains the degree-constrained minimum spanning tree problem as a subproblem.

16.36. Lagrangian decomposition (Guignard and Kim [1987a,b]). Consider the optimization problem P1 defined as $\min\{cx : \mathcal{A}x = b, \mathcal{D}x = d, x \geq 0 \text{ and integer}\}$. Suppose that by using a variable splitting technique described in Application 16.5, we restate this

problem in the following equivalent form P2: $\min\{\frac{1}{2}cx + \frac{1}{2}cy : \mathcal{A}x = b, \mathcal{D}y = d, x - y = 0, x, y \geq 0 \text{ and integer}\}$. We might form three different Lagrangian relaxations for this problem, one by relaxing the constraint $\mathcal{A}x = b$ in P2, one by relaxing the constraint $\mathcal{D}x = d$ in P2, and one by relaxing the constraint $x - y$ in P2. Let L^1, L^2, and L^3 denote the optimal values of the Lagrangian multiplier problems for each of these relaxations. The approach via problem L^3 is known as a *Lagrangian decomposition* since it permits us to decompose the problem into two separate subproblems, one corresponding to each set of equality constraints. Using Theorems 16.8 and Theorem 16.9, show that $L^3 \geq L^1$ and $L^3 \geq L^2$. (*Hint*: Let H^1 and H^2, respectively, denote the convex hulls of the sets $\{\mathcal{D}x = d, x \geq 0 \text{ and integer}\}$ and $\{\mathcal{A}x = b, x \geq 0 \text{ and integer}\}$. Consider the sets $H^1 \cap \{x : \mathcal{A}x = b\}$, $H^2 \cap \{x : \mathcal{D}x = d\}$, and $H^1 \cap H^2$, and consider the minimization of the objective function cx over each of these sets. Which of these problems has the smallest objective function value? What is the relationship between these optimal objective function values and the values L^1, L^2, and L^3?)

16.37. Example of Lagrangian decomposition. Suppose that we apply the Lagrangian decomposition procedure to the following integer programming example:

$$\text{Minimize} \quad -2x_1 - 3x_2$$

subject to

$$9x_1 + 10x_2 \leq 63,$$

$$4x_1 + 9x_2 \leq 36,$$

$$x_1, x_2 \geq 0 \text{ and integer.}$$

In this case, the reformulated problem is:

$$\text{Minimize} \quad -x_1 - \tfrac{3}{2}x_2 - y_1 - \tfrac{3}{2}y_2$$

subject to

$$9x_1 + 10x_2 \leq 63,$$

$$4y_1 + 9y_2 \leq 36,$$

$$x_1 - y_1 = 0,$$

$$x_2 - y_2 = 0,$$

$$x_1, x_2, y_1, y_2 \geq 0 \text{ and integer.}$$

Give a geometrical interpretation of the Lagrangian relaxation obtained by relaxing each of the following constraints: (1) $4x_1 + 9x_2 \leq 36$; (2) $4y_1 + 9y_2 \leq 36$; and (3) $x_1 - y_1 = 0$ and $x_2 - y_2 = 0$. From these geometrical considerations, interpret the fact that in the notation of Exercise 16.36, $L^3 \geq L^1$ and that $L^3 \geq L^2$.