inequalities is, in fact, $t_j \leq \frac{1}{2}T^*$; so adding the two inequalities gives us the bound

$$T_i \leq \frac{3}{2}T^*. \quad \blacksquare$$

11.2 The Center Selection Problem

Like the problem in the previous section, the Center Selection Problem, which we consider here, also relates to the general task of allocating work across multiple servers. The issue at the heart of Center Selection is where best to place the servers; in order to keep the formulation clean and simple, we will not incorporate the notion of load balancing into the problem. The Center Selection Problem also provides an example of a case in which the most natural greedy algorithm can result in an arbitrarily bad solution, but a slightly different greedy method is guaranteed to always result in a near-optimal solution.

The Problem

Consider the following scenario. We have a set S of n *sites*—say, n little towns in upstate New York. We want to select k *centers* for building large shopping malls. We expect that people in each of these n towns will shop at one of the malls, and so we want to select the sites of the k malls to be central.

Let us start by defining the input to our problem more formally. We are given an integer k, a set S of n sites (corresponding to the towns), and a distance function. When we consider instances where the sites are points in the plane, the distance function will be the standard Euclidean distance between points, and any point in the plane is an option for placing a center. The algorithm we develop, however, can be applied to more general notions of distance. In applications, distance sometimes means straight-line distance, but can also mean the travel time from point s to point z, or the driving distance (i.e., distance along roads), or even the cost of traveling. We will allow any distance function that satisfies the following natural properties.

- $dist(s, s) = 0$ for all $s \in S$
- the distance is symmetric: $dist(s, z) = dist(z, s)$ for all sites $s, z \in S$
- the triangle inequality: $dist(s, z) + dist(z, h) \geq dist(s, h)$

The first and third of these properties tend to be satisfied by essentially all natural notions of distance. Although there are applications with asymmetric distances, most cases of interest also satisfy the second property. Our greedy algorithm will apply to any distance function that satisfies these three properties, and it will depend on all three.

Next we have to clarify what we mean by the goal of wanting the centers to be "central." Let C be a set of centers. We assume that the people in a given town will shop at the closest mall. This suggests we define the distance of a site s to the centers as $dist(s, C) = \min_{c \in C} dist(s, c)$. We say that C forms an *r-cover* if each site is within distance at most r from one of the centers—that is, if $dist(s, C) \leq r$ for all sites $s \in S$. The minimum r for which C is an r-cover will be called the *covering radius* of C and will be denoted by $r(C)$. In other words, the covering radius of a set of centers C is the farthest that anyone needs to travel to get to his or her nearest center. Our goal will be to select a set C of k centers for which $r(C)$ is as small as possible.

Designing and Analyzing the Algorithm

Difficulties with a Simple Greedy Algorithm We now discuss greedy algorithms for this problem. As before, the meaning of "greedy" here is necessarily a little fuzzy; essentially, we consider algorithms that select sites one by one in a myopic fashion—that is, choosing each without explicitly considering where the remaining sites will go.

Probably the simplest greedy algorithm would work as follows. It would put the first center at the best possible location for a single center, then keep adding centers so as to reduce the covering radius, each time, by as much as possible. It turns out that this approach is a bit too simplistic to be effective: there are cases where it can lead to very bad solutions.

To see that this simple greedy approach can be really bad, consider an example with only two sites s and z, and $k = 2$. Assume that s and z are located in the plane, with distance equal to the standard Euclidean distance in the plane, and that any point in the plane is an option for placing a center. Let d be the distance between s and z. Then the best location for a single center c_1 is halfway between s and z, and the covering radius of this one center is $r(\{c_1\}) = d/2$. The greedy algorithm would start with c_1 as the first center. No matter where we add a second center, at least one of s or z will have the center c_1 as closest, and so the covering radius of the set of two centers will still be $d/2$. Note that the optimum solution with $k = 2$ is to select s and z themselves as the centers. This will lead to a covering radius of 0. A more complex example illustrating the same problem can be obtained by having two dense "clusters" of sites, one around s and one around z. Here our proposed greedy algorithm would start by opening a center halfway between the clusters, while the optimum solution would open a separate center for each cluster.

Knowing the Optimal Radius Helps In searching for an improved algorithm, we begin with a useful thought experiment. Suppose for a minute that someone told us what the optimum radius r is. Would this information help? That is, suppose we *know* that there is a set of k centers C^* with radius $r(C^*) \leq r$, and

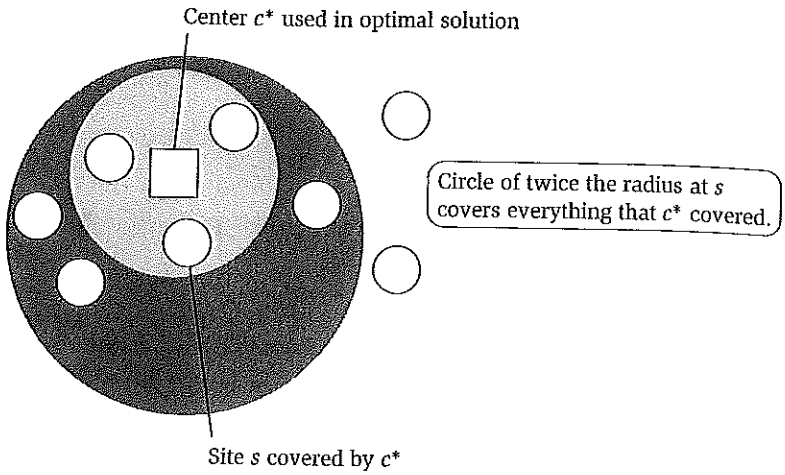

Figure 11.4 Everything covered at radius r by c^* is also covered at radius $2r$ by s.

our job is to find some set of k centers C whose covering radius is not much more than r. It turns out that finding a set of k centers with covering radius at most $2r$ can be done relatively easily.

Here is the idea: We can use the existence of this solution C^* in our algorithm even though we do not know what C^* is. Consider any site $s \in S$. There must be a center $c^* \in C^*$ that covers site s, and this center c^* is at distance at most r from s. Now our idea would be to take this site s as a center in our solution instead of c^*, as we have no idea what c^* is. We would like to make s cover all the sites that c^* covers in the unknown solution C^*. This is accomplished by expanding the radius from r to $2r$. All the sites that were at distance at most r from center c^* are at distance at most $2r$ from s (by the triangle inequality). See Figure 11.4 for a simple illustration of this argument.

S' will represent the sites that still need to be covered
Initialize $S' = S$
Let $C = \emptyset$
While $S' \neq \emptyset$
  Select any site $s \in S'$ and add s to C
  Delete all sites from S' that are at distance at most $2r$ from s
EndWhile
If $|C| \leq k$ then
  Return C as the selected set of sites
Else

```
    Claim (correctly) that there is no set of k centers with
       covering radius at most r
EndIf
```

Clearly, if this algorithm returns a set of at most k centers, then we have what we wanted.

(11.6) *Any set of centers C returned by the algorithm has covering radius $r(C) \leq 2r$.*

Next we argue that if the algorithm fails to return a set of centers, then its conclusion that no set can have covering radius at most r is indeed correct.

(11.7) *Suppose the algorithm selects more than k centers. Then, for any set C^* of size at most k, the covering radius is $r(C^*) > r$.*

Proof. Assume the opposite, that there is a set C^* of at most k centers with covering radius $r(C^*) \leq r$. Each center $c \in C$ selected by the greedy algorithm is one of the original sites in S, and the set C^* has covering radius at most r, so there must be a center $c^* \in C^*$ that is at most a distance of r from c—that is, $dist(c, c^*) \leq r$. Let us say that such a center c^* is *close* to c. We want to claim that no center c^* in the optimal solution C^* can be close to two different centers in the greedy solution C. If we can do this, we are done: each center $c \in C$ has a close optimal center $c^* \in C^*$, and each of these close optimal centers is distinct. This will imply that $|C^*| \geq |C|$, and since $|C| > k$, this will contradict our assumption that C^* contains at most k centers.

So we just need to show that no optimal center $c^* \in C$ can be close to each of two centers $c, c' \in C$. The reason for this is pictured in Figure 11.5. Each pair of centers $c, c' \in C$ is separated by a distance of more than $2r$, so if c^* were within a distance of at most r from each, then this would violate the triangle inequality, since $dist(c, c^*) + dist(c^*, c') \geq dist(c, c') > 2r$. ■

Eliminating the Assumption That We Know the Optimal Radius Now we return to the original question: How do we select a good set of k centers *without* knowing what the optimal covering radius might be?

It is worth discussing two different answers to this question. First, there are many cases in the design of approximation algorithms where it is conceptually useful to assume that you know the value achieved by an optimal solution. In such situations, you can often start with an algorithm designed under this assumption and convert it into one that achieves a comparable performance guarantee by simply trying out a range of "guesses" as to what the optimal

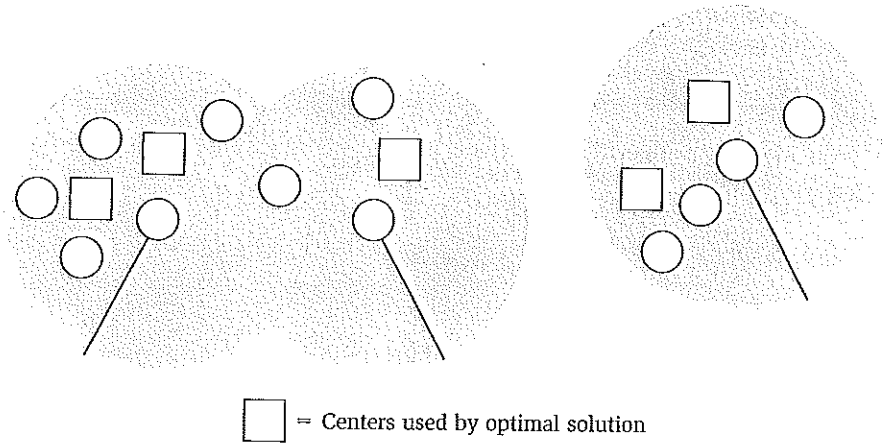

Figure 11.5 The crucial step in the analysis of the greedy algorithm that knows the optimal radius r. No center used by the optimal solution can lie in two different circles, so there must be at least as many optimal centers as there are centers chosen by the greedy algorithm.

value might be. Over the course of the algorithm, this sequence of guesses gets more and more accurate, until an approximate solution is reached.

For the Center Selection Problem, this could work as follows. We can start with some very weak initial guesses about the radius of the optimal solution: We know it is greater than 0, and it is at most the maximum distance r_{max} between any two sites. So we could begin by splitting the difference between these two and running the greedy algorithm we developed above with this value of $r = r_{max}/2$. One of two things will happen, according to the design of the algorithm: Either we find a set of k centers with covering radius at most $2r$, or we conclude that there is no solution with covering radius at most r. In the first case, we can afford to lower our guess on the radius of the optimal solution; in the second case, we need to raise it. This gives us the ability to perform a kind of binary search on the radius: in general, we will iteratively maintain values $r_0 < r_1$ so that we know the optimal radius is greater than r_0, but we have a solution of radius at most $2r_1$. From these values, we can run the above algorithm with radius $r = (r_0 + r_1)/2$; we will either conclude that the optimal solution has radius greater than $r > r_0$, or obtain a solution with radius at most $2r = (r_0 + r_1) < 2r_1$. Either way, we will have sharpened our estimates on one side or the other, just as binary search is supposed to do. We can stop when we have estimates r_0 and r_1 that are close to each other; at this point, our solution of radius $2r_1$ is close to being a 2-approximation to the optimal radius, since we know the optimal radius is greater than r_0 (and hence close to r_1).

A Greedy Algorithm That Works For the specific case of the Center Selection Problem, there is a surprising way to get around the assumption of knowing the radius, without resorting to the general technique described earlier. It turns out we can run essentially the same greedy algorithm developed earlier without knowing anything about the value of r.

The earlier greedy algorithm, armed with knowledge of r, repeatedly selects one of the original sites s as the next center, making sure that it is at least $2r$ away from all previously selected sites. To achieve essentially the same effect without knowing r, we can simply select the site s that is farthest away from all previously selected centers: If there is any site at least $2r$ away from all previously chosen centers, then this farthest site s must be one of them. Here is the resulting algorithm.

```
Assume k ≤ |S| (else define C = S)
Select any site s and let C = {s}
While |C| < k
   Select a site s ∈ S that maximizes dist(s, C)
   Add site s to C
EndWhile
Return C as the selected set of sites
```

(11.8) *This greedy algorithm returns a set C of k points such that $r(C) \leq 2r(C^*)$, where C^* is an optimal set of k points.*

Proof. Let $r = r(C^*)$ denote the minimum possible radius of a set of k centers. For the proof, we assume that we obtain a set of k centers C with $r(C) > 2r$, and from this we derive a contradiction.

So let s be a site that is more than $2r$ away from every center in C. Consider some intermediate iteration in the execution of the algorithm, where we have thus far selected a set of centers C'. Suppose we are adding the center c' in this iteration. We claim that c' is at least $2r$ away from all sites in C'. This follows as site s is more than $2r$ away from all sites in the larger set C, and we select a site c that is the farthest site from all previously selected centers. More formally, we have the following chain of inequalities:

$$dist(c', C') \geq dist(s, C') \geq dist(s, C) > 2r.$$

It follows that our greedy algorithm is a correct implementation of the first k iterations of the `while` loop of the previous algorithm, which knew the optimal radius r: In each iteration, we are adding a center at distance more than $2r$ from all previously selected centers. But the previous algorithm would

have $S' \neq \emptyset$ after selecting k centers, as it would have $s \in S'$, and so it would go on and select more than k centers and eventually conclude that k centers cannot have covering radius at most r. This contradicts our choice of r, and the contradiction proves that $r(C) \leq 2r$. ■

Note the surprising fact that our final greedy 2-approximation algorithm is a very simple modification of the first greedy algorithm that did not work. Perhaps the most important change is simply that our algorithm always selects sites as centers (i.e., every mall will be built in one of the little towns and not halfway between two of them).

11.3 Set Cover: A General Greedy Heuristic

In this section we will consider a very general problem that we also encountered in Chapter 8, the Set Cover Problem. A number of important algorithmic problems can be formulated as special cases of Set Cover, and hence an approximation algorithm for this problem will be widely applicable. We will see that it is possible to design a greedy algorithm here that produces solutions with a guaranteed approximation factor relative to the optimum, although this factor will be weaker than what we saw for the problems in Sections 11.1 and 11.2.

While the greedy algorithm we design for Set Cover will be very simple, the analysis will be more complex than what we encountered in the previous two sections. There we were able to get by with very simple bounds on the (unknown) optimum solution, while here the task of comparing to the optimum is more difficult, and we will need to use more sophisticated bounds. This aspect of the method can be viewed as our first example of the pricing method, which we will explore more fully in the next two sections.

The Problem

Recall from our discussion of NP-completeness that the Set Cover Problem is based on a set U of n elements and a list $S_1, \ldots, S_m$ of subsets of U; we say that a *set cover* is a collection of these sets whose union is equal to all of U.

In the version of the problem we consider here, each set S_i has an associated *weight* $w_i \geq 0$. The goal is to find a set cover $\mathcal{C}$ so that the total weight

$$\sum_{S_i \in \mathcal{C}} w_i$$

is minimized. Note that this problem is at least as hard as the decision version of Set Cover we encountered earlier; if we set all $w_i = 1$, then the minimum