algorithm returns a vertex cover whose size is at most twice the size of the maximal matching A. By relating the size of the solution returned to the lower bound, we obtain our approximation ratio. We will use this methodology in later sections as well.

Exercises

35.1-1
Give an example of a graph for which APPROX-VERTEX-COVER always yields a suboptimal solution.

35.1-2
Let A denote the set of edges that were picked in line 4 of APPROX-VERTEX-COVER. Prove that the set A is a maximal matching in the graph G.

35.1-3 ⋆
Professor Nixon proposes the following heuristic to solve the vertex-cover problem. Repeatedly select a vertex of highest degree, and remove all of its incident edges. Give an example to show that the professor's heuristic does not have an approximation ratio of 2. (*Hint:* Try a bipartite graph with vertices of uniform degree on the left and vertices of varying degree on the right.)

35.1-4
Give an efficient greedy algorithm that finds an optimal vertex cover for a tree in linear time.

35.1-5
From the proof of Theorem 34.12, we know that the vertex-cover problem and the NP-complete clique problem are complementary in the sense that an optimal vertex cover is the complement of a maximum-size clique in the complement graph. Does this relationship imply that there is a polynomial-time approximation algorithm with a constant approximation ratio for the clique problem? Justify your answer.

35.2 The traveling-salesman problem

In the traveling-salesman problem introduced in Section 34.5.4, we are given a complete undirected graph $G = (V, E)$ that has a nonnegative integer cost $c(u, v)$ associated with each edge $(u, v) \in E$, and we must find a hamiltonian cycle (a tour) of G with minimum cost. As an extension of our notation, let $c(A)$ denote the total cost of the edges in the subset $A \subseteq E$:

$$c(A) = \sum_{(u,v) \in A} c(u, v) .$$

In many practical situations, it is always cheapest to go directly from a place u to a place w; going by way of any intermediate stop v can't be less expensive. Putting it another way, cutting out an intermediate stop never increases the cost. We formalize this notion by saying that the cost function c satisfies the ***triangle inequality*** if for all vertices $u, v, w \in V$,

$$c(u, w) \leq c(u, v) + c(v, w) .$$

The triangle inequality is a natural one, and in many applications it is automatically satisfied. For example, if the vertices of the graph are points in the plane and the cost of traveling between two vertices is the ordinary euclidean distance between them, then the triangle inequality is satisfied. (There are many cost functions other than euclidean distance that satisfy the triangle inequality.)

As Exercise 35.2-2 shows, the traveling-salesman problem is NP-complete even if we require that the cost function satisfy the triangle inequality. Thus, it is unlikely that we can find a polynomial-time algorithm for solving this problem exactly. We therefore look instead for good approximation algorithms.

In Section 35.2.1, we examine a 2-approximation algorithm for the traveling-salesman problem with the triangle inequality. In Section 35.2.2, we show that without the triangle inequality, a polynomial-time approximation algorithm with a constant approximation ratio does not exist unless P = NP.

35.2.1 The traveling-salesman problem with the triangle inequality

Applying the methodology of the previous section, we will first compute a structure—a minimum spanning tree—whose weight is a lower bound on the length of an optimal traveling-salesman tour. We will then use the minimum spanning tree to create a tour whose cost is no more than twice that of the minimum spanning tree's weight, as long as the cost function satisfies the triangle inequality. The following algorithm implements this approach, calling the minimum-spanning-tree algorithm MST-PRIM from Section 23.2 as a subroutine.

```
APPROX-TSP-TOUR($G, c$)
1  select a vertex $r \in V[G]$ to be a "root" vertex
2  compute a minimum spanning tree $T$ for $G$ from root $r$
           using MST-PRIM($G, c, r$)
3  let $L$ be the list of vertices visited in a preorder tree walk of $T$
4  return the hamiltonian cycle $H$ that visits the vertices in the order $L$
```

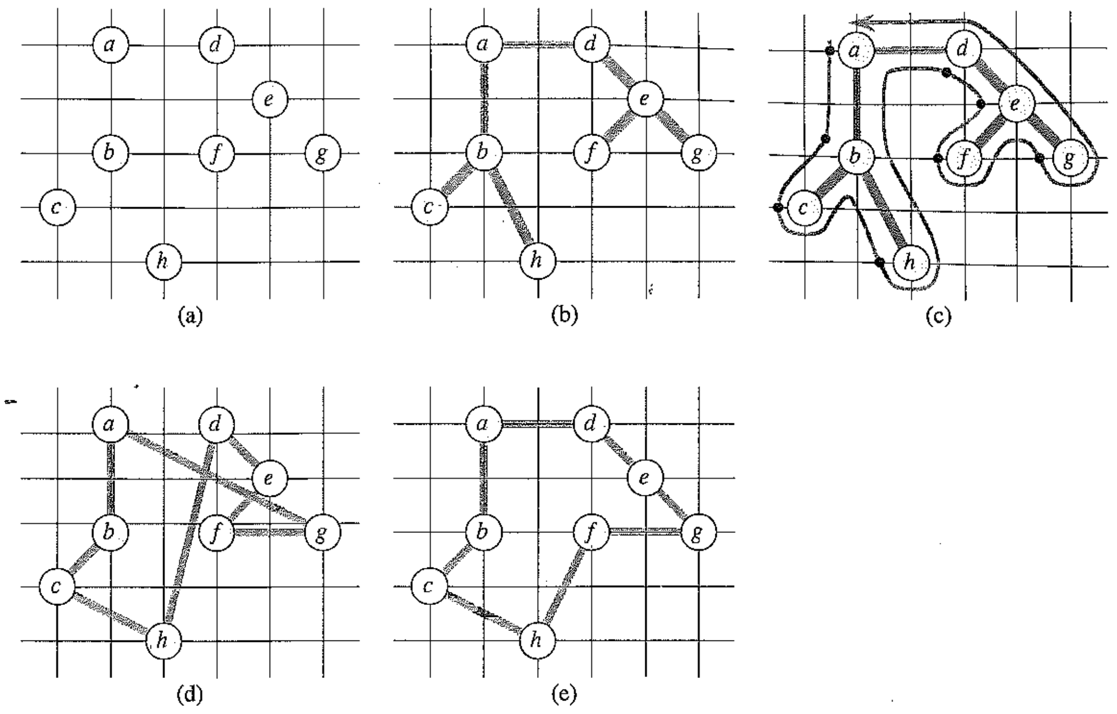

Figure 35.2 The operation of APPROX-TSP-TOUR. **(a)** The given set of points, which lie on vertices of an integer grid. For example, f is one unit to the right and two units up from h. The ordinary euclidean distance is used as the cost function between two points. **(b)** A minimum spanning tree T of these points, as computed by MST-PRIM. Vertex a is the root vertex. The vertices happen to be labeled in such a way that they are added to the main tree by MST-PRIM in alphabetical order. **(c)** A walk of T, starting at a. A full walk of the tree visits the vertices in the order $a, b, c, b, h, b, a, d, e, f, e, g, e, d, a$. A preorder walk of T lists a vertex just when it is first encountered, as indicated by the dot next to each vertex, yielding the ordering a, b, c, h, d, e, f, g. **(d)** A tour of the vertices obtained by visiting the vertices in the order given by the preorder walk. This is the tour H returned by APPROX-TSP-TOUR. Its total cost is approximately 19.074. **(e)** An optimal tour H^* for the given set of vertices. Its total cost is approximately 14.715.

Recall from Section 12.1 that a preorder tree walk recursively visits every vertex in the tree, listing a vertex when it is first encountered, before any of its children are visited.

Figure 35.2 illustrates the operation of APPROX-TSP-TOUR. Part (a) of the figure shows the given set of vertices, and part (b) shows the minimum spanning tree T grown from root vertex a by MST-PRIM. Part (c) shows how the vertices are visited by a preorder walk of T, and part (d) displays the corresponding tour, which is the tour returned by APPROX-TSP-TOUR. Part (e) displays an optimal tour, which is about 23% shorter.

By Exercise 23.2-2, even with a simple implementation of MST-PRIM, the running time of APPROX-TSP-TOUR is $\Theta(V^2)$. We now show that if the cost function for an instance of the traveling-salesman problem satisfies the triangle inequality, then APPROX-TSP-TOUR returns a tour whose cost is not more than twice the cost of an optimal tour.

Theorem 35.2
APPROX-TSP-TOUR is a polynomial-time 2-approximation algorithm for the traveling-salesman problem with the triangle inequality.

Proof We have already shown that APPROX-TSP-TOUR runs in polynomial time.

Let H^* denote an optimal tour for the given set of vertices. Since we obtain a spanning tree by deleting any edge from a tour, the weight of the minimum spanning tree T is a lower bound on the cost of an optimal tour, that is,

$$c(T) \le c(H^*) . \tag{35.4}$$

A ***full walk*** of T lists the vertices when they are first visited and also whenever they are returned to after a visit to a subtree. Let us call this walk W. The full walk of our example gives the order

$$a, b, c, b, h, b, a, d, e, f, e, g, e, d, a .$$

Since the full walk traverses every edge of T exactly twice, we have (extending our definition of the cost c in the natural manner to handle multisets of edges)

$$c(W) = 2c(T) . \tag{35.5}$$

Equations (35.4) and (35.5) imply that

$$c(W) \le 2c(H^*) , \tag{35.6}$$

and so the cost of W is within a factor of 2 of the cost of an optimal tour.

Unfortunately, W is generally not a tour, since it visits some vertices more than once. By the triangle inequality, however, we can delete a visit to any vertex from W and the cost does not increase. (If a vertex v is deleted from W between visits to u and w, the resulting ordering specifies going directly from u to w.) By repeatedly applying this operation, we can remove from W all but the first visit to each vertex. In our example, this leaves the ordering

$$a, b, c, h, d, e, f, g .$$

This ordering is the same as that obtained by a preorder walk of the tree T. Let H be the cycle corresponding to this preorder walk. It is a hamiltonian cycle, since every vertex is visited exactly once, and in fact it is the cycle computed by APPROX-TSP-TOUR. Since H is obtained by deleting vertices from the full walk W, we have

$$c(H) \le c(W) . \tag{35.7}$$

Combining inequalities (35.6) and (35.7) shows that $c(H) \le 2c(H^*)$, which completes the proof. ■

In spite of the nice approximation ratio provided by Theorem 35.2, APPROX-TSP-TOUR is usually not the best practical choice for this problem. There are other approximation algorithms that typically perform much better in practice. See the references at the end of this chapter.

35.2.2 The general traveling-salesman problem

If we drop the assumption that the cost function c satisfies the triangle inequality, good approximate tours cannot be found in polynomial time unless P = NP.

Theorem 35.3
If P $\neq$ NP, then for any constant $\rho \ge 1$, there is no polynomial-time approximation algorithm with approximation ratio ρ for the general traveling-salesman problem.

Proof The proof is by contradiction. Suppose to the contrary that for some number $\rho \ge 1$, there is a polynomial-time approximation algorithm A with approximation ratio ρ. Without loss of generality, we assume that ρ is an integer, by rounding it up if necessary. We shall then show how to use A to solve instances of the hamiltonian-cycle problem (defined in Section 34.2) in polynomial time. Since the hamiltonian-cycle problem is NP-complete, by Theorem 34.13, solving it in polynomial time implies that P = NP, by Theorem 34.4.

Let $G = (V, E)$ be an instance of the hamiltonian-cycle problem. We wish to determine efficiently whether G contains a hamiltonian cycle by making use of the hypothesized approximation algorithm A. We turn G into an instance of the traveling-salesman problem as follows. Let $G' = (V, E')$ be the complete graph on V; that is,

$$E' = \{(u, v) : u, v \in V \text{ and } u \neq v\} .$$

Assign an integer cost to each edge in E' as follows:

$$c(u, v) = \begin{cases} 1 & \text{if } (u, v) \in E , \\ \rho\,|V| + 1 & \text{otherwise} . \end{cases}$$