

35.2-3

Consider the following *closest-point heuristic* for building an approximate traveling-salesman tour whose cost function satisfies the triangle inequality. Begin with a trivial cycle consisting of a single arbitrarily chosen vertex. At each step, identify the vertex u that is not on the cycle but whose distance to any vertex on the cycle is minimum. Suppose that the vertex on the cycle that is nearest u is vertex v . Extend the cycle to include u by inserting u just after v . Repeat until all vertices are on the cycle. Prove that this heuristic returns a tour whose total cost is not more than twice the cost of an optimal tour.

35.2-4

In the *bottleneck traveling-salesman problem*, we wish to find the hamiltonian cycle that minimizes the cost of the most costly edge in the cycle. Assuming that the cost function satisfies the triangle inequality, show that there exists a polynomial-time approximation algorithm with approximation ratio 3 for this problem. (*Hint*: Show recursively that we can visit all the nodes in a bottleneck spanning tree, as discussed in Problem 23-3, exactly once by taking a full walk of the tree and skipping nodes, but without skipping more than two consecutive intermediate nodes. Show that the costliest edge in a bottleneck spanning tree has a cost that is at most the cost of the costliest edge in a bottleneck hamiltonian cycle.)

35.2-5

Suppose that the vertices for an instance of the traveling-salesman problem are points in the plane and that the cost $c(u, v)$ is the euclidean distance between points u and v . Show that an optimal tour never crosses itself.

35.3 The set-covering problem

The set-covering problem is an optimization problem that models many problems that require resources to be allocated. Its corresponding decision problem generalizes the NP-complete vertex-cover problem and is therefore also NP-hard. The approximation algorithm developed to handle the vertex-cover problem doesn't apply here, however, and so we need to try other approaches. We shall examine a simple greedy heuristic with a logarithmic approximation ratio. That is, as the size of the instance gets larger, the size of the approximate solution may grow, relative to the size of an optimal solution. Because the logarithm function grows rather slowly, however, this approximation algorithm may nonetheless give useful results.

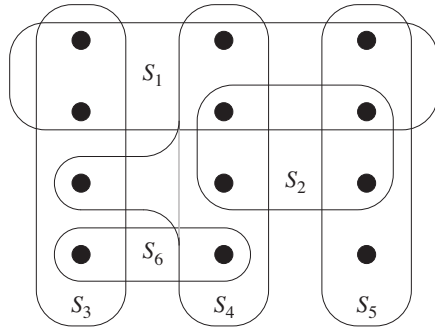


Figure 35.3 An instance (X, \mathcal{F}) of the set-covering problem, where X consists of the 12 black points and $\mathcal{F} = \{S_1, S_2, S_3, S_4, S_5, S_6\}$. A minimum-size set cover is $\mathcal{C} = \{S_3, S_4, S_5\}$, with size 3. The greedy algorithm produces a cover of size 4 by selecting either the sets S_1, S_4, S_5 , and S_3 or the sets S_1, S_4, S_5 , and S_6 , in order.

An instance (X, \mathcal{F}) of the *set-covering problem* consists of a finite set X and a family \mathcal{F} of subsets of X , such that every element of X belongs to at least one subset in \mathcal{F} :

$$X = \bigcup_{S \in \mathcal{F}} S.$$

We say that a subset $S \in \mathcal{F}$ *covers* its elements. The problem is to find a minimum-size subset $\mathcal{C} \subseteq \mathcal{F}$ whose members cover all of X :

$$X = \bigcup_{S \in \mathcal{C}} S. \quad (35.8)$$

We say that any \mathcal{C} satisfying equation (35.8) *covers* X . Figure 35.3 illustrates the set-covering problem. The size of \mathcal{C} is the number of sets it contains, rather than the number of individual elements in these sets, since every subset \mathcal{C} that covers X must contain all $|X|$ individual elements. In Figure 35.3, the minimum set cover has size 3.

The set-covering problem abstracts many commonly arising combinatorial problems. As a simple example, suppose that X represents a set of skills that are needed to solve a problem and that we have a given set of people available to work on the problem. We wish to form a committee, containing as few people as possible, such that for every requisite skill in X , at least one member of the committee has that skill. In the decision version of the set-covering problem, we ask whether a covering exists with size at most k , where k is an additional parameter specified in the problem instance. The decision version of the problem is NP-complete, as Exercise 35.3-2 asks you to show.

A greedy approximation algorithm

The greedy method works by picking, at each stage, the set S that covers the greatest number of remaining elements that are uncovered.

GREEDY-SET-COVER(X, \mathcal{F})

```

1   $U = X$ 
2   $\mathcal{C} = \emptyset$ 
3  while  $U \neq \emptyset$ 
4      select an  $S \in \mathcal{F}$  that maximizes  $|S \cap U|$ 
5       $U = U - S$ 
6       $\mathcal{C} = \mathcal{C} \cup \{S\}$ 
7  return  $\mathcal{C}$ 

```

In the example of Figure 35.3, GREEDY-SET-COVER adds to \mathcal{C} , in order, the sets S_1, S_4 , and S_5 , followed by either S_3 or S_6 .

The algorithm works as follows. The set U contains, at each stage, the set of remaining uncovered elements. The set \mathcal{C} contains the cover being constructed. Line 4 is the greedy decision-making step, choosing a subset S that covers as many uncovered elements as possible (breaking ties arbitrarily). After S is selected, line 5 removes its elements from U , and line 6 places S into \mathcal{C} . When the algorithm terminates, the set \mathcal{C} contains a subfamily of \mathcal{F} that covers X .

We can easily implement GREEDY-SET-COVER to run in time polynomial in $|X|$ and $|\mathcal{F}|$. Since the number of iterations of the loop on lines 3–6 is bounded from above by $\min(|X|, |\mathcal{F}|)$, and we can implement the loop body to run in time $O(|X| |\mathcal{F}|)$, a simple implementation runs in time $O(|X| |\mathcal{F}| \min(|X|, |\mathcal{F}|))$. Exercise 35.3-3 asks for a linear-time algorithm.

Analysis

We now show that the greedy algorithm returns a set cover that is not too much larger than an optimal set cover. For convenience, in this chapter we denote the d th harmonic number $H_d = \sum_{i=1}^d 1/i$ (see Section A.1) by $H(d)$. As a boundary condition, we define $H(0) = 0$.

Theorem 35.4

GREEDY-SET-COVER is a polynomial-time $\rho(n)$ -approximation algorithm, where $\rho(n) = H(\max\{|S| : S \in \mathcal{F}\})$.

Proof We have already shown that GREEDY-SET-COVER runs in polynomial time.

To show that GREEDY-SET-COVER is a $\rho(n)$ -approximation algorithm, we assign a cost of 1 to each set selected by the algorithm, distribute this cost over the elements covered for the first time, and then use these costs to derive the desired relationship between the size of an optimal set cover \mathcal{C}^* and the size of the set cover \mathcal{C} returned by the algorithm. Let S_i denote the i th subset selected by GREEDY-SET-COVER; the algorithm incurs a cost of 1 when it adds S_i to \mathcal{C} . We spread this cost of selecting S_i evenly among the elements covered for the first time by S_i . Let c_x denote the cost allocated to element x , for each $x \in X$. Each element is assigned a cost only once, when it is covered for the first time. If x is covered for the first time by S_i , then

$$c_x = \frac{1}{|S_i - (S_1 \cup S_2 \cup \dots \cup S_{i-1})|}.$$

Each step of the algorithm assigns 1 unit of cost, and so

$$|\mathcal{C}| = \sum_{x \in X} c_x. \quad (35.9)$$

Each element $x \in X$ is in at least one set in the optimal cover \mathcal{C}^* , and so we have

$$\sum_{S \in \mathcal{C}^*} \sum_{x \in S} c_x \geq \sum_{x \in X} c_x. \quad (35.10)$$

Combining equation (35.9) and inequality (35.10), we have that

$$|\mathcal{C}| \leq \sum_{S \in \mathcal{C}^*} \sum_{x \in S} c_x. \quad (35.11)$$

The remainder of the proof rests on the following key inequality, which we shall prove shortly. For any set S belonging to the family \mathcal{F} ,

$$\sum_{x \in S} c_x \leq H(|S|). \quad (35.12)$$

From inequalities (35.11) and (35.12), it follows that

$$\begin{aligned} |\mathcal{C}| &\leq \sum_{S \in \mathcal{C}^*} H(|S|) \\ &\leq |\mathcal{C}^*| \cdot H(\max\{|S| : S \in \mathcal{F}\}), \end{aligned}$$

thus proving the theorem.

All that remains is to prove inequality (35.12). Consider any set $S \in \mathcal{F}$ and any $i = 1, 2, \dots, |\mathcal{C}|$, and let

$$u_i = |S - (S_1 \cup S_2 \cup \dots \cup S_i)|$$

be the number of elements in S that remain uncovered after the algorithm has selected sets S_1, S_2, \dots, S_i . We define $u_0 = |S|$ to be the number of elements

of S , which are all initially uncovered. Let k be the least index such that $u_k = 0$, so that every element in S is covered by at least one of the sets S_1, S_2, \dots, S_k and some element in S is uncovered by $S_1 \cup S_2 \cup \dots \cup S_{k-1}$. Then, $u_{i-1} \geq u_i$, and $u_{i-1} - u_i$ elements of S are covered for the first time by S_i , for $i = 1, 2, \dots, k$. Thus,

$$\sum_{x \in S} c_x = \sum_{i=1}^k (u_{i-1} - u_i) \cdot \frac{1}{|S_i - (S_1 \cup S_2 \cup \dots \cup S_{i-1})|}.$$

Observe that

$$\begin{aligned} |S_i - (S_1 \cup S_2 \cup \dots \cup S_{i-1})| &\geq |S - (S_1 \cup S_2 \cup \dots \cup S_{i-1})| \\ &= u_{i-1}, \end{aligned}$$

because the greedy choice of S_i guarantees that S cannot cover more new elements than S_i does (otherwise, the algorithm would have chosen S instead of S_i). Consequently, we obtain

$$\sum_{x \in S} c_x \leq \sum_{i=1}^k (u_{i-1} - u_i) \cdot \frac{1}{u_{i-1}}.$$

We now bound this quantity as follows:

$$\begin{aligned} \sum_{x \in S} c_x &\leq \sum_{i=1}^k (u_{i-1} - u_i) \cdot \frac{1}{u_{i-1}} \\ &= \sum_{i=1}^k \sum_{j=u_i+1}^{u_{i-1}} \frac{1}{u_{i-1}} \\ &\leq \sum_{i=1}^k \sum_{j=u_i+1}^{u_{i-1}} \frac{1}{j} && \text{(because } j \leq u_{i-1}\text{)} \\ &= \sum_{i=1}^k \left(\sum_{j=1}^{u_{i-1}} \frac{1}{j} - \sum_{j=1}^{u_i} \frac{1}{j} \right) \\ &= \sum_{i=1}^k (H(u_{i-1}) - H(u_i)) \\ &= H(u_0) - H(u_k) && \text{(because the sum telescopes)} \\ &= H(u_0) - H(0) \\ &= H(u_0) && \text{(because } H(0) = 0\text{)} \\ &= H(|S|), \end{aligned}$$

which completes the proof of inequality (35.12). ■

Corollary 35.5

GREEDY-SET-COVER is a polynomial-time $(\ln |X| + 1)$ -approximation algorithm.

Proof Use inequality (A.14) and Theorem 35.4. ■

In some applications, $\max \{|S| : S \in \mathcal{F}\}$ is a small constant, and so the solution returned by GREEDY-SET-COVER is at most a small constant times larger than optimal. One such application occurs when this heuristic finds an approximate vertex cover for a graph whose vertices have degree at most 3. In this case, the solution found by GREEDY-SET-COVER is not more than $H(3) = 11/6$ times as large as an optimal solution, a performance guarantee that is slightly better than that of APPROX-VERTEX-COVER.

Exercises**35.3-1**

Consider each of the following words as a set of letters: {arid, dash, drain, heard, lost, nose, shun, slate, snare, thread}. Show which set cover GREEDY-SET-COVER produces when we break ties in favor of the word that appears first in the dictionary.

35.3-2

Show that the decision version of the set-covering problem is NP-complete by reducing it from the vertex-cover problem.

35.3-3

Show how to implement GREEDY-SET-COVER in such a way that it runs in time $O\left(\sum_{S \in \mathcal{F}} |S|\right)$.

35.3-4

Show that the following weaker form of Theorem 35.4 is trivially true:

$$|\mathcal{C}| \leq |\mathcal{C}^*| \max \{|S| : S \in \mathcal{F}\} .$$

35.3-5

GREEDY-SET-COVER can return a number of different solutions, depending on how we break ties in line 4. Give a procedure BAD-SET-COVER-INSTANCE(n) that returns an n -element instance of the set-covering problem for which, depending on how we break ties in line 4, GREEDY-SET-COVER can return a number of different solutions that is exponential in n .