Computer Science Department
University of Minnesota
Twin Cities
4-192 EE/CSci Building
200 Union Street S.E.
Minneapolis, MN 55455

On the Greedy Algorithm for a Covering Problem

by
Bhaskar Dasgupta, Ravi Janardan, and Naveed Sherwani

TR 93-13
February, 1993
Technical Report

On the Greedy Algorithm for a Covering Problem

Bhaskar Dasgupta* Ravi Janardan† Naveed Sherwani‡

February 11, 1993

Abstract

The following problem, called the *k-set cover problem*, is considered: Given a family of sets consisting of non-negatively weighted elements and an integer k, find a subfamily of k sets such that the sum of the weights of the elements in the union of the k sets is maximum. The k-set-cover problem is an abstraction of many commonly-arising combinatorial problems, such as, for instance, certain problems in facilities location and task assignment; however, it is NP-hard. Lower and upper bounds are established here on the quality of the approximation delivered by the greedy algorithm. Using the lower bound result, a simple proof is given of a well-known result concerning the performance of the greedy algorithm for the related minimum-set-cover problem. Also discussed is a specific example of the k-set-cover problem, namely, covering points in the plane with k axes-parallel rectangles.

Keywords: facilities location, greedy algorithm, rectangle cover, set cover.

*Dept. of Computer Science, University of Minnesota, Minneapolis, MN 55455. E-mail: dasgupta@cs.umn.edu. Research supported in part by NSF grant CCR-92-08913.

†Dept. of Computer Science, University of Minnesota, Minneapolis, MN 55455. E-mail: janardan@cs.umn.edu. Research supported in part by NSF grant CCR-92-00270.

‡Dept. of Computer Science, Western Michigan University, Kalamazoo, MI 49008. E-mail: sherwani@cs.wmich.edu. Research supported in part by NSF grant USE-90-52346.

1 Introduction

We consider the following problem, which we call the *k-set-cover* problem: Let $\mathcal{U} = \{u_1, u_2, \ldots, u_n\}$ be a set of n elements and let $w(u_i) \geq 0$ be a *weight* associated with each $u_i \in \mathcal{U}$. Let $\mathcal{F} = \{F_1, F_2, \ldots, F_m\}$ be a family of m subsets of $\mathcal{U}$. Given an integer k, where $1 \leq k \leq m$, we wish to find a subfamily $\mathcal{A} = \{A_1, A_2, \ldots, A_k\}$ of $\mathcal{F}$ such that $w(\mathcal{A}) = \sum_{u \in \bigcup A_j} w(u)$ is maximized. (Note that $\mathcal{F}$ need not be known explicitly; it could be defined implicitly by specifying a property on $\mathcal{U}$, as in the example below.)

The k-set-cover problem can be viewed as an abstraction of the following problem in geometric location theory: Suppose that we are given a set of n *sites* (points) in $\mathcal{R}^d$, $d \geq 1$. We wish to locate k *facilities* (points) such that we maximize the number of sites that are within a specified distance, r, from at least one facility. (Here, r is measured in the L_t-metric for some $t \geq 1$.[1]) In other words, we wish to locate k points in $\mathcal{R}^d$ such that we maximize the number of distinct sites that are covered by (i.e., contained in) the L_t-balls of radius r centered at these points.[2] We can cast this location problem as a k-set-cover problem by taking $\mathcal{U}$ to be the set of sites, setting $w(u_i) = 1$ for each $u_i \in \mathcal{U}$, and defining $\mathcal{F}$ to be the collection of those subsets F_i of $\mathcal{U}$ such that the sites in F_i are all coverable by an L_t-ball of radius r centered at some point in $\mathcal{R}^d$. (Thus $\mathcal{F}$ is specified implicitly.) Then each subset $A_j \in \mathcal{A}$ represents the sites that are covered by some L_t-ball in the optimal placement of the k facilities. (Usually, the process of computing the A_j's will also yield the actual placement of the facilities.) More generally, each site could have a weight associated with it (e.g., a population count) and the goal could be to locate the k facilities so as to maximize the total weight of the sites covered, rather than the number of sites covered.

In fact, by defining $\mathcal{F}$ suitably, we can extend the above interpretation to (i) covering by k copies of an arbitrary shape (i.e, not just an L_t-ball), (ii) covering by k copies drawn from a set of more than one shape (e.g., circles of different radii,

[1]If $p = (p_1, p_2, \ldots, p_d)$ and $q = (q_1, q_2, \ldots, q_d)$ are points in $\mathcal{R}^d$, then the *L_t-distance* between p and q is $(\sum_{i=1}^{d} |p_i - q_i|^t)^{1/t}$ if $1 \leq t < \infty$. The *L_∞-distance* between p and q is $\max_{1 \leq i \leq d} |p_i - q_i|$.

[2]In $\mathcal{R}^d$, the *L_t-ball of radius r centered at a point* p is the set of all points q such that the L_t-distance between p and q is at most r. For example, in $\mathcal{R}^2$, such a ball is a circle of radius r and center p if $t = 2$, a square of side $\sqrt{2}r$ centered at p and tilted 45° if $t = 1$, and an axes-parallel square of side $2r$ centered at p if $t = \infty$.

rectangles of different dimensions, or even combinations of circles, rectangles etc.), and (iii) covering objects other than points (e.g., line segments). Other interpretations of the k-set-cover problem, for example as a task-assignment problem (see [CLR90]), are also possible. Thus, due to its many applications, efficient algorithms for the k-set-cover problem are of considerable interest.

Unfortunately, however, the k-set-cover problem is NP-hard. This follows from the NP-completeness of the well-known *minimum-set-cover* problem [CLR90, GJ79]:

"Given a set $\mathcal{U} = \{u_1, u_2, \ldots, u_n\}$, a family $\mathcal{F} = \{F_1, F_2, \ldots, F_m\}$ of subsets of $\mathcal{U}$ such that each u_i is in at least one F_j, and an integer q, is there a subfamily $\mathcal{F}'$ of $\mathcal{F}$ consisting of at most q sets such that each $u_i \in \mathcal{U}$ is in at least one $F_j \in \mathcal{F}'$?"

Clearly, the answer to the set-cover problem is 'yes' if and only if $w(\mathcal{A}) = n$ for the corresponding q-set-cover problem in which $w(u_i) = 1$ for all $u_i \in \mathcal{U}$. In fact, even the problem of covering n points in $\mathcal{R}^2$ with k axes-parallel rectangles is NP-hard, if k is part of the input [FPT81].

A natural approach for the k-set-cover problem is the *greedy algorithm*, which iteratively picks subsets $G_1, G_2, \ldots, G_k$ of $\mathcal{F}$ (in that order) as follows: For any subset U of $\mathcal{U}$, let the *weight of U* be $w(U) = \sum_{u \in U} w(u)$. Then, for $1 \leq i \leq k$, G_i is the subset of $\mathcal{F}$ for which $w(G_i - \bigcup_{j=1}^{i-1} G_j)$ is maximum. That is, G_1 is a maximum weight subset of $\mathcal{F}$ and, for $i > 1$, G_i is a subset of $\mathcal{F}$ for which the total weight of the elements that have not yet been covered is maximum. If $T(n,m)$ is the time to pick a maximum-weight subset of $\mathcal{F}$, then the greedy algorithm runs in time $O(kT(n,m))$.

As an example, consider the problem of covering the maximum number of sites in $\mathcal{R}^2$ by k axes-parallel rectangles. In the greedy algorithm, we would find the rectangle which covers the maximum number of sites, delete the covered sites, and repeat the process $k-1$ times. Since a rectangle which covers the maximum number of sites can be found in time $T(n,m) = T(n) = O(n \log n)$ [IA83], the greedy algorithm runs in $O(kn \log n)$ time. Similarly, for the problem of covering with circles, the greedy algorithm takes $O(kn^2)$ time, since a circle covering the maximum number of sites can be found in time $O(n^2)$ [CL86].

How good is the solution delivered by the greedy algorithm? In this paper, we give lower and upper bounds on the performance of the greedy algorithm, as stated in the following theorem:

Theorem 1 *Let $\mathcal{U} = \{u_1, u_2, \ldots, u_n\}$ be a set of n elements and let $w(u_i) \geq 0$ be the weight of $u_i \in \mathcal{U}$. Let $\mathcal{F} = \{F_1, F_2, \ldots, F_m\}$ be a family of m subsets of $\mathcal{U}$. For any integer $k \geq 1$, let $\mathcal{A} = \{A_1, A_2, \ldots, A_k\}$ be a subfamily of $\mathcal{F}$ such that $w(\mathcal{A}) = \sum_{u \in \bigcup A_j} w(u)$ is maximum. Let $\mathcal{G} = \{G_1, G_2, \ldots, G_k\}$ be the solution produced by the greedy algorithm and let $w(\mathcal{G}) = \sum_{u \in \bigcup G_i} w(u)$. Let $\rho_k = w(\mathcal{G})/w(\mathcal{A})$. Then*

(a) $\rho_k \geq \max\{\frac{3}{k+2}, \frac{1}{2} + \frac{1}{4k-2}\}$ *for any problem instance. Thus, for example, $\rho_2 = 0.75$, $\rho_3 = 0.6$, $\rho_4 = 0.57$ etc., and ρ_k approaches 0.5 for large k.*

(b) *for each k, there exists a problem instance for which $\rho_k = 1 - (1 - \frac{1}{k})^k$. Thus, for example, $\rho_2 = 0.75$, $\rho_3 = 0.7$, $\rho_4 = 0.68$ etc., and for large k, ρ_k approaches $1 - \frac{1}{e} \approx 0.63$, where $e = 2.718...$ is the base of natural logarithms.* □

We will give a proof of this result in Sections 2 and 3. Using part (a) of Theorem 1, we also give, in Section 4, an alternative proof of the well-known result [Chv79, CLR90] that the greedy algorithm for the optimization version of the minimum-set-cover problem has a performance ratio of $O(\log n)$. (In the optimization version of the minimum-set-cover problem, we wish to find the smallest number of subsets of $\mathcal{F}$ that cover all the elements of $\mathcal{U}$. The greedy algorithm for this problem works by repeatedly finding a subset of $\mathcal{F}$ which covers the maximum number of elements not yet covered.) In Section 5, we discuss the problem of covering points in the plane with k axes-parallel rectangles. We conclude in Section 6 with some open problems.

2 Proof of Theorem 1(a)

Let $G'_i = G_i - \bigcup_{j=1}^{i-1} G_j$ and let $g_i = w(G'_i)$, for $1 \leq i \leq k$. Let $A'_j = A_j - \bigcup_{i=1}^{j-1} A_i$ and let $a_j = w(A'_j)$, for $1 \leq j \leq k$. That is, g_i is the total weight of the elements covered by G_i but not covered by any of $G_1, G_2, \ldots, G_{i-1}$. Similarly, a_j is the total weight of the elements covered by A_j but not covered by any of $A_1, A_2, \ldots, A_{j-1}$.

We note the following: (i) $g_1 \geq g_2 \geq \ldots \geq g_k$ by definition of the greedy method. Moreover, the sets in $\mathcal{A}$ can always be indexed such that $a_1 \geq a_2 \geq \ldots \geq a_k$; thus we can assume without loss of generality that this is the case. (ii) Any two of the G'_i's are disjoint and so $w(G'_1 \cup \ldots \cup G'_i) = g_1 + \cdots + g_i$, for $1 \leq i \leq k$. Likewise, any two of the A'_l's are disjoint and so $w(A'_1 \cup \ldots \cup A'_j) = a_1 + \cdots + a_j$, for $1 \leq j \leq k$. (iii) $\bigcup_{i=1}^{l} G_i = \bigcup_{i=1}^{l} G'_i$ and $\bigcup_{j=1}^{l} A_j = \bigcup_{j=1}^{l} A'_j$, for $1 \leq l \leq k$; thus, in particular, $w(\mathcal{G}) = g_1 + \cdots + g_k$ and $w(\mathcal{A}) = a_1 + \cdots + a_k$.

We now prove a useful lemma.

Lemma 1 *Let $a_1, \ldots, a_k$ and $g_1, \ldots, g_k$ be as defined above. Then, for $1 \leq i \leq k$, $g_1 + g_2 + \ldots + g_{i-1} \geq a_1 + a_2 + \ldots + a_k - kg_i$.*

Proof Consider the time at which G_i is selected. We claim that $G'_1 \cup \ldots \cup G'_{i-1}$ covers elements of A'_j of total weight at least $a_j - g_i$, i.e., $w((G'_1 \cup \ldots \cup G'_{i-1}) \cap A'_j) \geq a_j - g_i$ for all j, $1 \leq j \leq k$.

Suppose for a contradiction that the claim is false. Thus, there exists a set A_h, where $1 \leq h \leq k$, such that $w((G'_1 \cup \ldots \cup G'_{i-1}) \cap A'_h) < a_h - g_i$. Then, by selecting A_h as our G_i, we have $g_i = w(A_h - \bigcup_{l=1}^{i-1} G_l) \geq w(A'_h - \bigcup_{l=1}^{i-1} G_l) = w(A'_h - \bigcup_{l=1}^{i-1} G'_l) > a_h - (a_h - g_i) = g_i$, a contradiction.

Since any two sets A'_p and A'_q, where $p \neq q$, are disjoint, $(G'_1 \cup \ldots \cup G'_{i-1}) \cap A'_p)$ and $(G'_1 \cup \ldots \cup G'_{i-1}) \cap A'_q)$ are also disjoint. This and the above claim imply that $w(G'_1 \cup \ldots \cup G'_{i-1}) \geq (a_1 - g_i) + \cdots + (a_k - g_i)$. Thus $g_1 + g_2 + \ldots + g_{i-1} \geq a_1 + a_2 + \ldots + a_k - kg_i$. □

Proof of $\rho_k \geq \frac{3}{k+2}$

By definition of the greedy method, we have $g_1 \geq a_1$. Let $\alpha > 1$ be a real-valued parameter whose value will be fixed later. We divide the analysis into two parts:

Case 1: $g_i \geq a_i/\alpha$, for $2 \leq i \leq k$.

We then have

$$\begin{aligned} \rho_k &= (g_1 + \cdots + g_k)/(a_1 + \cdots + a_k) \\ &\geq (a_1 + \frac{1}{\alpha}(a_2 + \cdots + a_k))/(a_1 + \cdots a_k) \\ &= 1 - \frac{\alpha - 1}{\alpha}(a_2 + \cdots + a_k)/(a_1 + \cdots a_k) \\ &\geq 1 - \frac{\alpha - 1}{\alpha}\frac{k-1}{k} \qquad (1) \end{aligned}$$

The last step follows from the fact that $a_1 \geq \frac{1}{k}(a_1 + \cdots + a_k)$, since it is the maximum of the a_i; thus $a_2 + \cdots + a_k \leq (1 - \frac{1}{k})(a_1 + \cdots + a_k)$.

Case 2: $g_i \geq a_i/\alpha$, for $2 \leq i < j$, and $g_j < a_j/\alpha$, where $2 \leq j \leq k$. (We do not care about the g_i for $i > j$.)

By Lemma 1, applied to the selection of G_j, we have $g_1 + \cdots + g_{j-1} \geq a_1 + \cdots + a_k - kg_j$. Thus $g_1 + \cdots + g_k \geq g_1 + \cdots g_{j-1} + g_j \geq a_1 + \cdots + a_k - (k-1)g_j$.

We now have:

$$\begin{aligned}
\rho_k &\geq (a_1 + \cdots + a_k - (k-1)g_j)/(a_1 + \cdots a_k) \\
&= 1 - (k-1)g_j/(a_1 + \cdots + a_k) \\
&> 1 - (k-1)\frac{a_j}{\alpha}/(a_1 + \cdots + a_k) \\
&\geq 1 - (k-1)\frac{a_j}{\alpha}/(a_1 + \cdots + a_j) \\
&\geq 1 - (k-1)\frac{a_j}{\alpha}/ja_j \quad \text{because } a_1 \geq a_2 \cdots \geq a_j \\
&= 1 - (k-1)/j\alpha \\
&\geq 1 - (k-1)/2\alpha \qquad (2)
\end{aligned}$$

Equating (1) and (2) we get $\alpha = \frac{k}{2} + 1$. Thus $\rho_k \geq \frac{3}{k+2}$.

Proof of $\rho_k \geq \frac{1}{2} + \frac{1}{4k-2}$

For $k \geq 4$, we can strengthen the lower bound on ρ_k from $\frac{3}{k+2}$ to $\frac{1}{2} + \frac{1}{4k-2}$, as follows.

Each G'_i covers some elements (possibly none) of each of $A'_1, A'_2, \ldots, A'_k$ that are not covered by any of $G'_1, G'_2, \ldots, G'_{i-1}$. Denote the total weights of these elements by $w_{i1}, w_{i2}, \ldots, w_{ik}$, respectively. Additionally, G'_i may cover elements that are not in any of the A''s. Denote the total weight of these elements by x_i. Hence, $g_i = x_i + \sum_{j=1}^{k} w_{ij}$ and

$$\begin{aligned}
w(\mathcal{G}) &= \sum_{i=1}^{k}(x_i + \sum_{j=1}^{k} w_{ij}) \\
&= X + \sum_{i=1}^{k}\sum_{j=1}^{k} w_{ij},
\end{aligned}$$

where $X = \sum_{i=1}^{k} x_i$. Also, each A'_j may cover elements that are not covered by any of the G'_i. Denote the total weight of these elements by y_j. Hence, $a_j = y_j + \sum_{i=1}^{k} w_{ij}$ and

$$\begin{aligned} w(\mathcal{A}) &= \sum_{j=1}^{k}(y_j + \sum_{i=1}^{k} w_{ij}) \\ &= \sum_{j=1}^{k}\sum_{i=1}^{k} w_{ij}, \end{aligned}$$

where $Y = \sum_{j=1}^{k} y_j$. Thus,

$$\rho_k = \frac{w(\mathcal{G})}{w(\mathcal{A})} = \frac{w(\mathcal{G})}{w(\mathcal{G}) - X + Y} = \frac{1}{1 + \frac{Y-X}{w(\mathcal{G})}}.$$

Consider the time at which G_i is selected, $1 \le i \le k$. The total weight of the elements of A'_j, $1 \le j \le k$, that are not covered by $G'_1 \cup \ldots \cup G'_{i-1}$ is $\sum_{l=i}^{k} w_{lj} + y_j$. By definition of the greedy method, we have

$$y_j + \sum_{l=i}^{k} w_{lj} \le g_i, \quad \text{for } 1 \le j \le k. \tag{3}$$

Summing up (3) over all j we have

$$\begin{aligned} Y + \sum_{j=1}^{k}\sum_{l=i}^{k} w_{lj} &\le kg_i \\ \text{i.e., } Y + \sum_{l=i}^{k}\sum_{j=1}^{k} w_{lj} &\le kg_i \\ \text{i.e., } Y + \sum_{l=i}^{k}(g_l - x_l) &\le kg_i \\ \text{i.e., } Y - \sum_{l=i}^{k} x_l &\le (k-1)g_i - \sum_{l=i+1}^{k} g_l \\ \text{i.e., } Y - X &\le (k-1)g_i, \quad \text{for } 1 \le i \le k. \end{aligned} \tag{4}$$

Summing up (4) over all i, we have

$$k(Y - X) \leq (k-1)\sum_{i=1}^{k} g_i = (k-1)w(\mathcal{G}).$$

Thus, $\frac{Y-X}{w(\mathcal{G})} \leq \frac{k-1}{k}$ and we get

$$\rho_k \geq \frac{1}{1+\frac{k-1}{k}} = \frac{k}{2k-1} = \frac{1}{2} + \frac{1}{4k-2}.$$

Combining this with the previous bound we have

$$\rho_k \geq \max\left\{\frac{3}{k+2}, \frac{1}{2} + \frac{1}{4k-2}\right\},$$

which concludes the proof of Theorem 1(a).

3 Proof of Theorem 1(b)

For any $k > 1$, we will construct an instance of the k-set-cover problem such that $\rho_k = 1 - (1 - \frac{1}{k})^k$. (We omit discussion of the case $k = 1$ since, by definition of the greedy method, $\rho_1 = 1$ for any problem instance.)

Consider the following matrix $U = (u_{ij})$, where $0 \leq i \leq k$ and $1 \leq j \leq k$. In what follows, $\beta > 1$ is a parameter to be fixed later.

$$\begin{array}{lll}
u_{01} = 1 & u_{0j} = \frac{k-2}{k-1} & 2 \leq j \leq k \\
u_{11} = 0 & u_{1j} = \frac{1}{k-1} & 2 \leq j \leq k \\
u_{ij} = \frac{1}{k}\beta^{i-1} & & 2 \leq i \leq k, 1 \leq j \leq k
\end{array}$$

The desired k-set-cover instance is as follows: Let $\mathcal{U} = \{u_{ij} \mid 0 \leq i \leq k, 1 \leq j \leq k\}$ and let $w(u_{ij}) = u_{ij}$. Let $\mathcal{F} = \{R_1, \ldots, R_k, C_1, \ldots, C_k\}$, where $R_i = \{u_{ij} \mid 1 \leq j \leq k\}$ is the ith row of U and $C_j = \{u_{ij} \mid 0 \leq i \leq k\}$ is the jth column.

The optimal cover is $\mathcal{A} = \{C_1, \ldots, C_k\}$, which covers all the elements of $\mathcal{U}$ and has total weight $w(\mathcal{A}) = 1 + (k-2) + 1 + \sum_{i=2}^{k} \beta^{i-1} = (k-1) + \sum_{i=0}^{k-1} \beta^i$. We claim that β can be chosen such that the greedy algorithm selects the sets $R_k, R_{k-1}, \ldots, R_1$ (in that order). The total weight of $\mathcal{G} = \{R_k, \ldots, R_1\}$ is $w(\mathcal{G}) = \sum_{i=0}^{k-1} \beta^i$. Thus,

$$\rho_k = \frac{w(\mathcal{G})}{w(\mathcal{A})} = \frac{1}{1 + \frac{k-1}{\sum_{i=0}^{k-1}\beta^i}} = \frac{1}{1 + \frac{(k-1)(\beta-1)}{\beta^k - 1}} \tag{5}$$

We now demonstrate the existence of a suitable β. Consider the instant just before R_{k-i+1} is selected, where $i = 1, 2, \ldots, k$. Since the R_l's are all disjoint, the total weight of the elements covered by R_{k-i+1} but not by any previously-chosen R is just $w(R_{k-i+1})$. Let $C'_{ij} \subseteq C_j$ be the elements of C_j that have not been covered by the R's chosen so far, where $1 \leq j \leq k$. For the selection of $R_k, \ldots, R_1$ to be greedy, we must have: (i) $w(R_{k-i+1}) \geq w(R_{k-l+1})$, where $1 \leq i \leq k-1$ and $i < l \leq k$ and (ii) $w(R_{k-i+1}) \geq w(C'_{ij})$, where $1 \leq i \leq k$ and $1 \leq j \leq k$.

Consider condition (i). We have $w(R_{k-i+1}) = k\frac{1}{k}\beta^{k-i} = \beta^{k-i}$. Likewise, we have $w(R_{k-l+1}) = \beta^{k-l}$ (this formula holds for $l = k$ also since $w(R_1) = (k-1)\frac{1}{k-1} = 1$). Since $\beta > 1$ and $l > i$, we have $w(R_{k-i+1}) > w(R_{k-l+1})$ and so (i) holds.

Consider condition (ii). If $i = k$, then $w(R_1) = 1$ and $w(C'_{kj}) = \frac{k-2}{k-1} + \frac{1}{k-1} = 1$ for $1 \leq j \leq k$. Thus (ii) holds in this case. Now consider any $i < k$. For $1 \leq j \leq k$, we have

$$w(C'_{ij}) = \frac{k-2}{k-1} + \frac{1}{k-1} + \sum_{t=1}^{k-i} \frac{1}{k}\beta^t = 1 + \frac{\beta}{k}\frac{\beta^{k-i}-1}{\beta-1}.$$

For (ii) to hold, we want

$$\begin{aligned} \beta^{k-i} &\geq 1 + \frac{\beta}{k}\frac{\beta^{k-i}-1}{\beta-1} \\ \text{i.e., } \beta^{k-i} - 1 &\geq \frac{\beta}{k}\frac{\beta^{k-i}-1}{\beta-1} \\ \text{i.e., } 1 &\geq \frac{\beta}{k}\frac{1}{\beta-1}, \quad \text{since } \beta^{k-i} - 1 > 0 \\ \text{i.e., } \beta &\geq \frac{k}{k-1} \end{aligned}$$

To make ρ_k in (5) as small as possible, we must choose β as small as possible. Thus, we choose $\beta = \frac{k}{k-1}$. Substituting this into (5) and simplifying, we get

$$\rho_k = 1 - \left(1 - \frac{1}{k}\right)^k.$$

Since $(1 - \frac{1}{k})^k = (1 - \frac{1}{k}) \cdot \frac{1}{(1+\frac{1}{k-1})^{k-1}}$ and $\lim_{x\to\infty}(1 + \frac{1}{x})^x = e = 2.718...$—the base of natural logarithms—it follows that ρ_k approaches $1 - \frac{1}{e}$ for large k.

4 Performance bound of the greedy algorithm for the minimum-set-cover problem

As mentioned in the Introduction, it is well-known (see [CLR90, Chv79]) that for the optimization version of the minimum-set-cover problem, the number of sets used by the greedy algorithm to cover $\mathcal{U}$ is $O(\log n)$ times larger than the number of sets used by the optimal solution. In this section, we show how this result can be derived easily once Theorem 1(a) (or any result which gives a constant lower bound on ρ_k) is known.

Suppose that the optimal solution to a given instance of the minimum-set-cover problem uses q sets. We claim that the first q (or fewer) sets picked by the greedy algorithm cover at least $n/2$ elements of $\mathcal{U}$. To see this, consider the q-set-cover problem for this instance, with $w(u) = 1$ for each $u \in \mathcal{U}$. Clearly, the optimal solution to this instance covers n elements of $\mathcal{U}$. Thus, by Theorem 1(a), the greedy algorithm for this instance of the q-set-cover problem covers more than $n/2$ elements of $\mathcal{U}$. This proves the claim since the greedy algorithm is the same for the minimum-set-cover problem and for the q-set-cover problem.

Thus the number of elements of $\mathcal{U}$ that have not been covered by the first q (or fewer) greedy sets is $n' < n/2$. Consider now the instance of the minimum-set-cover problem obtained by restricting the original instance to the n' non-covered elements, i.e., by deleting the covered elements from $\mathcal{U}$ and from the sets in $\mathcal{F}$. Clearly, the optimal solution to this instance uses at most q sets. Arguing as before, we can show that the next q (or fewer) greedy sets cover at least $n'/2$ elements of $\mathcal{U}$.

Thus the number of elements of $\mathcal{U}$ that are not covered by the first $2q$ (or fewer) greedy sets is less than $n' - n'/2 = n'/2 < n/2^2$. Inductively, the number of elements of $\mathcal{U}$ that are not covered by the first jq (or fewer) greedy sets is less than $n/2^j$. It follows that the first $O(\log n)$ greedy sets suffice to cover all the elements of $\mathcal{U}$, which establishes the desired performance bound for the minimum-set-cover problem.

5 The k-rectangle-cover problem

This problem is a specific example of the k-set-cover problem and is defined as follows: We are given a set, S, of n nonnegatively-weighted points in the plane and a set, $\mathcal{T}$, of t different templates of axes-parallel rectangles. The goal is to maximize the total weight of the points covered by k rectangles drawn from $\mathcal{T}$ (the same template can be used more than once).

We first note that if k is part of the input, then the problem is NP-hard, even if $t = 1$ and the points all have unit weight. This follows from the NP-completeness of the corresponding *minimum-rectangle-cover* problem [FPT81], where we have to decide if a set S of n points in the plane can be covered by q or fewer identical, axes-parallel rectangles, for any specified integer q. Clearly, the answer to this problem is 'yes' if and only if for the corresponding q-set-cover problem, with all points assigned unit weight, the total weight of the points covered is n.

For fixed $k \geq 1$, however, the k-set-cover problem can be solved in $O(t^k n^{2k-1} \log n)$ time, as follows: First, note that for $k = 1$ the problem is solvable in $O(tn \log n)$ time, since we can determine the optimal solution for each template in $\mathcal{T}$ in $O(n \log n)$ time [IA83]. Now consider the case $k = 2$. Observe that there is always an optimal solution in which the left side and the bottom side of one of the rectangles contains a point of S (the two points could be the same, i.e., they could coincide with the southwest corner of the rectangle). This is so because given any optimal solution we can always translate rightwards one of the rectangles until its left side encounters a point of S and then translate it upwards until its bottom side encounters a point of S; clearly, this does not affect the total weight of the points covered. Thus, there are $O(tn^2)$ choices for one of the rectangles in an optimal solution. For each such choice, we solve the 1-rectangle-cover problem on the points not covered. Thus the total time is $O(t^2 n^3 \log n)$. Inductively, for $k > 2$, we can solve the k-rectangle-cover problem by trying each of the possible $O(tn^2)$ choices for one of the rectangles in an optimal solution and then solving recursively, in $O(t^{k-1} n^{2(k-1)-1} \log n)$ time, the $(k-1)$-rectangle-cover problem for the points not covered. The total time is thus $O(t^k n^{2k-1} \log n)$.

The above algorithm is expensive even for small k. However, by using the greedy algorithm to repeatedly solve (for each template) the 1-rectangle-cover problem on the points not yet covered, we get a simple algorithm whose running time is just $O(ktn \log n)$ and whose performance is guaranteed by Theorem 1(a).

For $t = 2$, the upper bound on ρ_k given by Theorem 1(b) can be translated directly to the k-rectangle-cover problem by positioning $n = k^2 + k$ points on a $(k+1) \times k$ grid and letting the rectangle templates be a $(k+1) \times 1$ rectangle and a $1 \times k$ rectangle.

The upper bound of Theorem 1(b) does not hold for $t = 1$ (i.e., when we are covering with k identical, axes-parallel rectangles). For this case, we can establish upper bounds for any k as follows: For $k = 2$, the example in Figure 1(a) provides an upper bound of $\frac{6+\epsilon}{8} = \frac{3}{4} + \epsilon'$ on ρ_k, where $\epsilon > 0$ and $\epsilon' > 0$ are vanishingly small. For $k = 3$, the example in Figure 1(b) provides an upper bound of $\frac{7+\epsilon}{9} = \frac{7}{9} + \epsilon'$ on ρ_k.

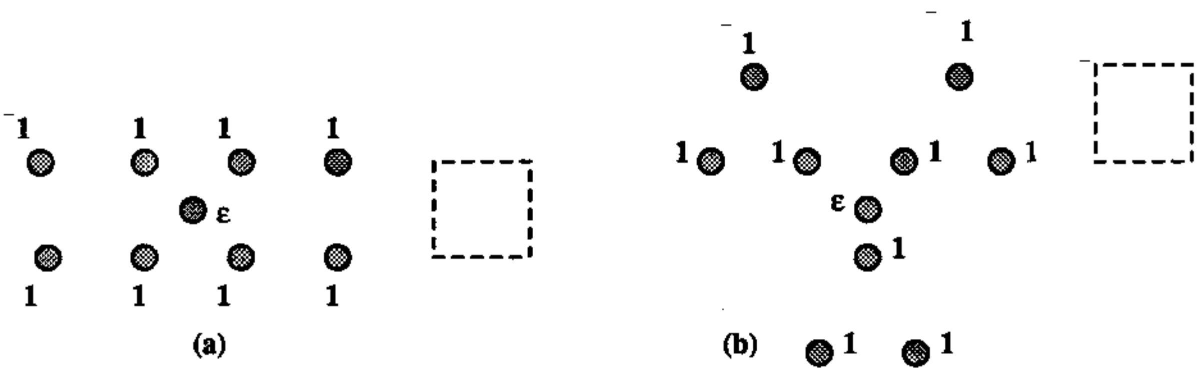

Figure 1: *Examples realizing the upper bound when only one rectangle template (shown dashed) is allowed. The weight of each point is shown beside it. (a) $k = 2$; $\rho_2 = \frac{3}{4} + \epsilon'$. (b) $k = 3$; $\rho_3 = \frac{7}{9} + \epsilon'$. Here $\epsilon > 0$ and $\epsilon' > 0$ are vanishingly small.*

If $k > 2$ is an even number, then we can simply use $\frac{k}{2}$ copies of the example in Figure 1(a), all sufficiently far apart. This gives an upper bound of $\frac{(6+\epsilon)\frac{k}{2}}{8\frac{k}{2}} = \frac{3}{4} + \epsilon'$ on ρ_k. If $k > 3$ is an odd number, then we simply use $\frac{k-3}{2}$ copies of the example in Figure 1(a) and one copy of the example in Figure 1(b), all far apart. This gives an upper bound on ρ_k of $\frac{(6+\epsilon)\frac{k-3}{2}+7+\epsilon}{8\frac{k-3}{2}+9} = \frac{3}{4} + \frac{1}{16k-12} + \epsilon'$.

6 Conclusion

We have given almost-matching lower and upper bounds on the performance of the greedy algorithm for the k-set-cover problem, which is an abstraction of many commonly-arising combinatorial problems but is NP-hard.

Several interesting open problems remain: (i) Can the gap between the lower and upper bounds in Theorem 1 be narrowed further? (ii) Is there an efficient approximation algorithm (perhaps a polynomial-time approximation scheme) which performs better than the greedy algorithm? (iii) Can fast, exact algorithms be derived for special cases, such as, for instance, covering points in the plane with $k > 1$ identical, axes-parallel rectangles (or circles), for fixed k? And (iv) can sharper bounds be established on the performance of the greedy algorithm for covering points in the plane with $k > 1$ identical, axes-parallel rectangles?

References

[Chv79] V. Chvátal. A greedy heuristic for the set-covering problem. *Mathematics of Operations Research*, 4:233–235, 1979.

[CL86] B.M. Chazelle and D.T. Lee. On a circle placement problem. *Computing*, 36:1–16, 1986.

[CLR90] T.H. Cormen, C.E. Leiserson, and R.L. Rivest. *Introduction to Algorithms.* MIT Press and McGraw-Hill, 1990.

[FPT81] R.J. Fowler, M.S. Paterson, and S.L. Tanimoto. Optimal packing and covering in the plane are NP-complete. *Information Processing Letters*, 12:133–137, 1981.

[GJ79] M.R. Garey and D.S. Johnson. *Computers and Intractability—A guide to the theory of NP-Completeness.* W.H. Freeman, 1979.

[IA83] H. Imai and T. Asano. Finding the connected components and a maximum clique of an intersection graph of rectangles in the plane. *Journal of Algorithms*, 4:310–323, 1983.