Analysis of the Greedy Approach in Problems of Maximum *k*-Coverage

Dorit S. Hochbaum,[1] Anu Pathria[2]

[1] *Department of Industrial Engineering and Operations Research and Walter A. Haas School of Business, University of California, Berkeley, California, 94720-1777, USA*

[2] *HNC Software Inc., 5930 Cornerstone Court West, San Diego, California, 92121-3728, USA*

Received June 1995; revised March 1998; accepted 9 March 1998

Abstract: In this paper, we consider a general covering problem in which k subsets are to be selected such that their union covers as large a weight of objects from a universal set of elements as possible. Each subset selected must satisfy some structural constraints. We analyze the quality of a k-stage covering algorithm that relies, at each stage, on greedily selecting a subset that gives maximum improvement in terms of overall coverage. We show that such greedily constructed solutions are guaranteed to be within a factor of $1 - 1/e$ of the optimal solution. In some cases, selecting a best solution at each stage may itself be difficult; we show that if a β-approximate best solution is chosen at each stage, then the overall solution constructed is guaranteed to be within a factor of $1 - 1/e^{\beta}$ of the optimal. Our results also yield a simple proof that the number of subsets used by the greedy approach to achieve entire coverage of the universal set is within a logarithmic factor of the optimal number of subsets. Examples of problems that fall into the family of general covering problems considered, and for which the algorithmic results apply, are discussed. © 1998 John Wiley & Sons, Inc. Naval Research Logistics 45: 615–627, 1998

INTRODUCTION

Covering problems, in which a universal set of elements is to be covered with as few subsets as possible, where each subset must satisfy some structural constraints, arise in many contexts. Many of these problems are known to be NP-complete. The focus of this paper is on developing approximation algorithms for covering problems in which the objective is to select a *fixed* number of subsets k such that maximum coverage of the universal set is achieved. Since we do not require that all elements of the universal set be covered, it makes sense to assign weights to the individual elements; maximum coverage corresponds to selecting k subsets that cover a set of elements with total weight as great as possible. Because the problem of selecting a *minimum* number of subsets that cover *all* elements in

Correspondence to: D. S. Hochbaum
Contract grant sponsor: Office of Naval Research; contract grant number: N00014-91-J-1241
Contract grant sponsor: National Science Foundation; contract grant number: DMI-9713482

© 1998 by John Wiley & Sons, Inc. CCC 0894-069X/98/060615-13

$U' \leftarrow U$
for $l = 1 \ldots$ k do
select $A_l \in \mathcal{R}$ such that $A_l \subseteq U'$
$U' \leftarrow U' - A_l$
end
$A(k) \leftarrow [A_1, A_2, \ldots, A_k]$
output $A(k)$

Figure 1. Generic *k*-stage covering algorithm.

a universal set is NP-hard, so too is the problem of covering a *maximum* set of elements with a *fixed* number of subsets.

We derive results for a greedy-like approximation algorithm for such covering problems in a very general setting so that, while the details vary from problem to problem, the results regarding the quality of solution returned apply in a general way. Johnson [17] and Chvátal [5] have analyzed such an algorithm for the related problem of optimally covering all elements in a universal set. In this paper, we have attempted to abstract away as many details of the covering problem we consider as possible, and to make our algorithm as generic as possible, so that our analysis is applicable to a wide variety of problems.

The maximum *k*-coverage problem we consider can be described as follows:

INSTANCE: A universal set of elements U, an integer k, and a class $\mathcal{R}$ of subsets of U. Each element $u \in U$ has an associated weight $w(u)$.

OPTIMIZATION PROBLEM: Select k subsets, $A_1, A_2 \ldots, A_k$, of U, where each subset selected is a member of $\mathcal{R}$, such that the weight of the elements in $U_{l=1}^{k} A_l$ is maximized.

We assume, without loss of generality, that $A \in \mathcal{R}$ implies that $A' \in \mathcal{R}$ for all $A' \subseteq A$. So, if there is an element in both $A_i \in \mathcal{R}$ and $A_j \in \mathcal{R}$, then it can be removed from A_j say (A_j remains in $\mathcal{R}$), without changing the quality of the solution to the covering problem. Hence, we can add the restriction to the general covering problem that $A_i \cap A_j = \emptyset$, for all $1 \leq i < j \leq k$; that is, no two subsets selected have any elements in common. This restriction will make our notation, and subsequent presentation, cleaner; an alternative would have been to specify that, after a subset A_i is selected for the cover, all remaining subsets which have those elements that are also in A_i removed. Beyond this nonrestrictive assumption that the selected subsets are mutually disjoint, it is important to note that there are no conditions that the A_i's must satisfy with respect to each other.

In Section 2, a generic *k*-stage algorithm that selects the subsets $A_1, A_2, \ldots, A_k$ in an iterative manner is given, and performance guarantees are derived for greedy-like implementations of the algorithm. Section 3 develops approximation algorithms for a variety of problems and applications for which the results of Section 2 apply. Problems, taken from a variety of applications, regarding covering graphs with subgraphs, packing, and fixed parameter combinatorial optimization are considered. In Section 4, we extend the analysis of Section 2 in several ways, and then conclude with a summary.

1. A *k*-STAGE ALGORITHM

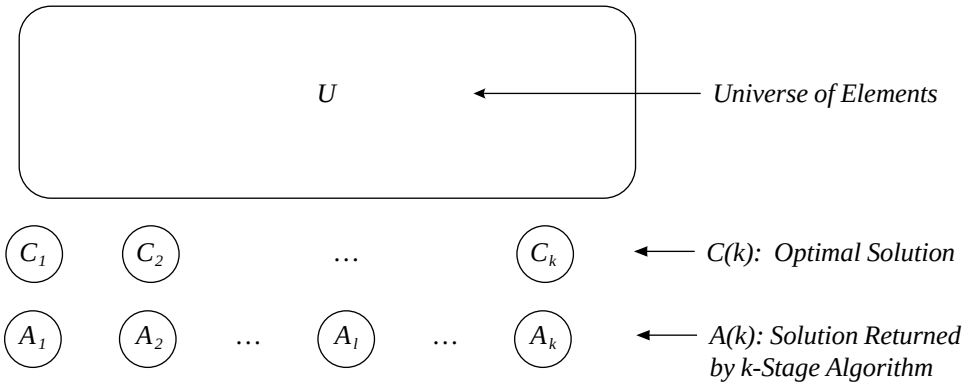
Consider the generic *k*-stage covering algorithm, shown in Figure 1, that selects a subset A_l at stage l. (Note that, either because there are fewer than k subsets in $\mathcal{R}$ or because of

our assumption that selected sets are disjoint, it may be that at some stage l no additional set can be selected. In such a situation, the solution obtained after stage $l-1$ will provide optimal coverage.)

Let $C(k) = [C_1, C_2, \ldots, C_k]$ be an optimal solution, where $c(k)$ is the *value* (weight of elements covered) of the optimal solution. For $l = 1, 2, \ldots, k$, let $A(l) = [A_1, A_2, \ldots, A_l]$ be the solution constructed up to the end of stage l of the generic algorithm; so, $A(k)$ is the final solution returned. We denote the value of A_l by a_l, and the value of $A(l)$ by $a(l)$:

$$a_l = \sum_{u \in A_l} w(u) \quad \text{and} \quad a(l) = \sum_{i=1}^{l} a_i = \sum_{i=1}^{l} \sum_{u \in A_i} w(u).$$

So, $a(k)$ is the value of the solution returned by the algorithm:



1.1. Greedy and Pseudo-Greedy Implementations

Consider a greedy implementation of the generic k-stage covering algorithm that selects A_l at stage l of maximum weight; that is, the subset that provides the maximum improvement in terms of coverage is chosen. Using this implementation, we will show in Corollary 1 that

$$\frac{a(k)}{c(k)} > 1 - \frac{1}{e}.$$

One practical problem with the greedy implementation is that at each step we wish to find a subset of maximum weight. This may itself be NP-hard. Suppose, however, if instead of selecting a maximum weight subset at each stage, we select a subset guaranteed to have weight within a factor of β of the optimal. We provide an analysis to establish the quality of the solution generated by this *pseudo-greedy* implementation; namely, we show (Theorem 1) that

$$\frac{a(k)}{c(k)} > 1 - \frac{1}{e^{\beta}}.$$

The quality of the greedy implementation is achieved by setting $\beta = 1$.

We begin with a couple of lemmas.

LEMMA 1: $a_l \geq \beta\,[c(k) - a(l-1)]/k$, for $l = 1, 2, \ldots, k$.

PROOF: At least $c(k) - a(l-1)$ worth of elements not covered by $A(l-1)$ are covered by the k subsets of $C(k)$. Hence, by the pigeonhole principle, one of the k subsets in the optimal solution must cover at least $[c(k) - a(l-1)]/k$ worth of these elements. Since A_l is a β-approximation to the maximum weight subset available, the result follows. □

LEMMA 2: $a(l) \geq (1 - (1 - \beta/k)^l)c(k)$, for $l = 1, 2, \ldots, k$.

PROOF: Proceed by induction on l. For $l = 1$, the result holds: $a(1) = a_1 \geq \beta\, c(k)/k$, from Lemma 1. Now,

$$a(l+1) = a(l) + a_{l+1} \geq a(l) + \beta\,\frac{c(k) - a(l)}{k}$$

$$= \left(1 - \frac{\beta}{k}\right)a(l) + \beta\,\frac{c(k)}{k} \geq \left(1 - \frac{\beta}{k}\right)\left(1 - \left(1 - \frac{\beta}{k}\right)^l\right)c(k) + \beta\,\frac{c(k)}{k}$$

$$= \left(1 - \left(1 - \frac{\beta}{k}\right)^{l+1}\right)c(k),$$

where the first inequality comes from Lemma 1, and the second inequality is from the induction hypothesis. □

The main result of this subsection follows directly from Lemma 2:

THEOREM 1: $a(k) \geq \alpha_k(\beta)c(k)$, where $\alpha_k(\beta) = 1 - (1 - \beta/k)^k > 1 - 1/e^\beta$.

PROOF: First of all,

$$a(k) \geq (1 - (1 - \beta/k)^k)c(k) = \alpha_k(\beta)c(k),$$

from Lemma 2. Secondly, because $\lim_{k\to\infty} \alpha_k(\beta) = 1 - 1/e^\beta$ and $\alpha_k(\beta)$ is decreasing, it follows that $\alpha_k(\beta) > 1 - 1/e^\beta$. □

Thus, the pseudo-greedy implementation provides an $\alpha_k(\beta)$-approximate solution to the covering problem.

COROLLARY 1: For the greedy implementation, $a(k) \geq \alpha_k(1)c(k) = \alpha_k c(k)$, where $\alpha_k = 1 - (1 - 1/k)^k > 1 - 1/e$.

PROOF: Set $\beta = 1$. □

Thus, the greedy implementation provides an α_k-approximate solution to the covering problem. In Section 3, we provide a problem instance that shows this bound to be tight.

The following table gives an indication of the quality of solution that is guaranteed to be returned by the greedy implementation of the *k*-stage covering algorithm, for particular k:

k	α_k
1	1.000
2	.750
3	.703
4	.683
5	.672
10	.651
100	.633
∞	.632

2. APPLICATIONS

We now turn to problem applications that fall into the family of general covering problems we have been considering. Approximation algorithms, with performance guarantees using the results of the previous section, are developed.

2.1. Covering Graphs by Subgraphs

Covering Edges by Cut Sets

Consider the problem of covering the (possibly weighted) edges of a graph, G, by a minimum cardinality set of cuts. This problem has been raised in the context of testing printed circuit boards for short-circuits (see [18]). The graph represents a printed circuit board, with nodes corresponding to connection points and edges corresponding to components. The components that can be tested for short-circuits in one single testing stage must all belong to some single cut in G: The electric testing is performed by connecting some connection points (nodes) to a + charge and the other connection points to a − charge, with those components whose two connection points have opposite charge being tested. The goal is to test as many components as possible in k stages.

We consider the problem of selecting k subsets of edges, where the edges in any particular subset must belong to the same cut, such that the maximum weight set of edges in G are covered. We would like to select, at each stage of the *k*-stage covering algorithm, a cut of maximum weight (in the graph consisting of edges not already covered). Now, although *MAX CUT* is itself NP-hard (see [9]), there is a simple greedy *MAX CUT* algorithm with $\beta = \frac{1}{2}$, and a more involved one with

$$\beta = \min_{0 \le \theta \le \pi} \frac{2}{\pi} \frac{\theta}{1 - \cos \theta} = .878 \cdots$$

(see [11]).

The following table provides an indication of how good various implementations of the

pseudo-greedy k-stage covering algorithm are (or how bad, depending on your perspective) for these two values of β [1]:

k	$\alpha_k(.878)$	$\alpha_k(.5)$
1	.878	.500
2	.685	.437
3	.646	.421
4	.628	.413
5	.619	.409
10	.601	.401
100	.585	.394
∞	.584	.393

There is also a relationship between this problem and graph coloring (see [2, 20]); indeed, this relationship establishes that the problem discussed above is NP-hard. Consider a binary code of length k assigned to each node, where the lth digit indicates which side of the lth cut the node is in. If each distinct code corresponds to a particular color (hence, there are 2^k colors), then the edges of the graph that correspond to tested components are precisely the edges that are bicolored (that is, edges whose end points have different colors); a similar relationship can be established in the reverse direction, where you begin with a coloring. The result that follows from this relationship is that, for a given set of edges E', if χ is the minimum number of vertex colors required to bicolor E' and k is the minimum number of cut-sets required to cover E', then $k = \lceil \log_2 \chi \rceil$. So, suppose that we wish to assign 2^k colors to the nodes of G so that as many edges as possible are bicolored. Then, the greedy algorithm as given can be used to derive a coloring that is within a factor of $\alpha_k(\beta)$ of the optimal, where β represents the quality of the approximation algorithm for *Max Cut.*

Other Edge Covering Problems

Problems of covering the edges of a graph by subgraphs satisfying some particular structure arise in other contexts as well. For example, we may wish to cover the maximum weight set of edges in a graph G using k subgraphs from a class $\mathcal{R}$; $\mathcal{R}$ may consist of triangles or other small cliques (see [8]), or spanning trees, etc. At each stage, an optimal structure from the class $\mathcal{R}$ is selected in the graph that is identical to G except that previously covered edges have weight 0. If $\mathcal{R}$ consists of triangles or spanning trees, an α_k-approximate solution can be found in polynomial time, because optimal spanning trees and triangles can be found in polynomial time. We note that efficient algorithms are given in [8] for covering *all* the edges of G with a number of cliques that is within $\frac{7}{5}$, $\frac{7}{3}$, and $\frac{3}{4}k$ of the optimal, where the size of the cliques is no greater than 3, 4, and k, respectively.

Covering Vertices with Independent Sets

Consider a k-stage forestry problem in which a set of cells are to be harvested at each stage, under the restriction that no two adjacent cells can be harvested during a given stage;

[1] It has been pointed out [10] that for this problem of covering edges with k cut sets, employing a randomized algorithm based on the method of conditional expectations provides a performance guarantee of $1 - 1/2^k$.

that is, at each stage an *Independent Set* set must be selected in a corresponding graph (see [3]). The goal in the *k*-stage problem may be to cover the maximum weight set of vertices in a graph G (which corresponds to the cell structure in the harvesting problem) using k subsets, where each subset must form an independent set in the graph. If $k \geq \chi(G)$, where $\chi(G)$ is the chromatic number of the graph G, then the optimal solution covers the entire set of edges, as each color in an optimal coloring forms an independent set. We wish to find a maximum weight independent set, at each stage, in the subgraph of G induced by those vertices not in any previously selected independent set. While the independent set problem is NP-hard for general graphs (indeed, guaranteeing a β-approximate solution for any fixed $\beta > 0$ is NP-hard), for certain classes of graphs (such as planar, sparse, bounded degree, etc.) approximation algorithms of varying quality are available (see [1, 13, 14, 19]). We point out that Barahona, Weintraub, and Epstein [3] give no performance guarantee for their approach.

2.2. Packing and Layout Problems

Consider now packing problems in which you have a set U of objects to pack. The common nature of these applications is that the objective is to pack the maximum weight set of objects into k identical *bins.* (Contrast this to the usual bin packing problems in which the goal is to find the *fewest* number of bins such that *all* items are packed.) We give examples in this subsection from logistics, VLSI design, and scheduling in which such packing problems arise. Depending on the application, there will be a description of the set U, a description of a bin, and certain constraints restricting which sets of objects can be packed into a common bin. Our *k*-stage greedy approach is to pack, as best possible, a single bin at a time using objects of U not already packed.

Circuit Layout and Design

Recent advances in multilayer IC technology have led to design problems in which an optimal assignment of objects to layers is to be made. For example, it has been shown that the topological planar routing problem, in which a maximum weight set of nets is to be assigned to k given layers such that all nets assigned to a given layer can be routed without any two nets crossing each other, is NP-hard (see [6]). On the other hand, finding a maximum weighted subset of nets to assign to a single layer can be solved in polynomial time (see [22]). It follows that our greedy approach yields a polynomial time α_k-approximation algorithm to the topological planar routing problem. We hope to explore further design problems arising as a result of the advancement in multilayer technology (for example, see [16]).

Scheduling

Suppose we are given a set of jobs to assign to k identical machines, where each machine has a set of restrictions as to which jobs can be grouped together (for example, exclusion of certain job groupings, deadline constraints, etc.). The goal is to schedule the largest weight set of jobs to the k machines. How well a single machine can be packed, in a stage of our greedy approach, will depend on the restrictions as to what can be scheduled together on a machine and depend on the set of jobs to be scheduled. Thus, if we can identify a schedule for a single machine that is within β of the optimum, then the resulting schedule

for the k machines is within $\alpha_k(\beta)$. See [12] for a survey of approximation algorithms for scheduling problems.

Logistics

Consider a logistics problem in which k identical vehicles are to be packed with a maximum weight set of items for delivery to a common destination. Given a set of items to be delivered, each having specified benefit, we attempt to pack the vehicles, one at a time, with items of maximum total benefit. Depending on the nature of both the vehicle and the items to be packed, finding the optimal assignment of items to the next vehicle may itself be a difficult problem. Packing a single vehicle within β of the optimum, results in a packing of the k vehicles within $\alpha_k(\beta)$.

In summary, each stage of the packing process for the problems above requires an optimal set of objects to be selected for the next bin. Depending on the types of objects to be packed (size, shape, etc.), and the description of the bin (shape, bad blocks, etc.), a variety of approximation algorithms to pack a single bin exist (for instance, see [13]).

2.3. Fixed Parameter Combinatorial Optimization

We now consider some additional classic optimization problems in which we wish to cover all the elements in a set using the smallest number of subsets. In our variations, we can look at these problems in which, given a fixed parameter k that limits the number of subsets that we can select, we wish to cover the maximum weight set of elements. Each of these problems uses a very simple procedure at each stage that greedily finds the best subset possible. Consider, for example, *Dominating Set:*

DOMINATING SET: We wish to ''dominate'' the maximum weight set of vertices in a graph G using k vertices. At each stage, we select a vertex that covers the maximum weight set of vertices not previously covered.

Other such α_k-approximation algorithms can be derived for fixed parameter versions of well-known combinatorial optimization problems such as *Vertex Cover, Set Cover, Minimum Test Set, Hitting Set,* and *Minimum Test Collection.* Refer to [9] for complete specifications of these problems.

We note that several location problems, in which the goal is to locate k facilities so that as many customers as possible can each be served within a prespecified cost, can be modeled as a fixed-parameter version of the *Dominating Set* problem. For example, a problem in which the goal is to locate k new facilities so as to maximize market share (see [21]), can be modeled as such; while the results in [21] give special cases of the problem that can be solved in polynomial time, our greedy approach provides an α_k-approximation algorithm for general instances of the problem considered. We also note that a similar problem concerning the optimal location of bank accounts was given in [4]; indeed, they provide a greedy algorithm that is shown to have a α_k-approximation guarantee via a different, and more complicated, analysis than the one presented in this paper.

3. ADDITIONAL ANALYSIS OF THE GREEDY APPROACH

In this section, we first give an instance of the general covering problem that establishes the tightness of the α_k bound for the greedy algorithm derived in Section 2. We then extend

the analysis of Section 2 in two distinct ways: first, we quantify the quality of the solution obtained from running the *k*-stage algorithm on an instance in terms of the quality of the solution actually found at each stage; second, we analyze the quality of solution obtained, with respect to the optimal solution based on *k* subsets, if one is allowed to select more than *k* subsets. As a corollary to the latter extension, we establish that the greedy approach can be used to cover the entire set U using a number of subsets that is within a logarithmic factor of the optimal.

3.1. Tightness of α_k Bound for Greedy Approach

This presentation is based on a result given in [7]. Dasgupta, Janardan, and Sherwani [7] exhibited a family of instances of the fixed parameter *Set Cover* problem for which a greedy approach (which is equivalent to the greedy implementation of our *k*-stage covering algorithm) yields a solution that is exactly α_k times that of the optimal.

For $k = 1$, $\alpha_k = 1$, so that the greedy implementation of the *k*-stage algorithm yields an optimal solution. For k an integer greater than 1, consider the following instance of the generic covering problem, in which k subsets are to be selected.

$U : U = \{(i, j) | 0 \le i \le k, 1 \le j \le k\}$. U can be thought of as a matrix of elements, with weights $u_{i,j}$:

$$\text{Row 0:} \quad u_{0,j} = \begin{cases} k - 1, & \text{if } j = 1, \\ k - 2, & 2 \le j \le k, \end{cases}$$

$$\text{Row 1:} \quad u_{1,j} = \begin{cases} 0, & \text{if } j = 1, \\ 1, & 2 \le j \le k, \end{cases}$$

$$\text{Row } i\text{:} \quad u_{i,j} = \left(\frac{k}{k-1}\right)^{i-2}, \quad 2 \le i \le k, 1 \le j \le k.$$

$\mathcal{R}$: $\mathcal{R}$ consists of those subsets of U such that all of the elements in the subset are either in the same column of U or in the same row (with the exception of row 0) of U.

Let C_i denote the subset consisting of the elements of the ith column of U, and let R_j denote the subset consisting of the elements of the jth row of U. Clearly, an optimal solution is given by $C(k) = [C_1, C_2, \ldots, C_k]$, as all of the elements of U are covered. Now, it follows from the following two claims that the greedy approach may yield a solution with weight exactly α_k times that of the optimal.

CLAIM 1: The greedy implementation of the *k*-stage covering algorithm can select the solution $[R_k, R_{k-1}, \ldots, R_1]$.

CLAIM 2: The weight of the elements in rows 1 through k of U equals α_k times that of all the elements in U.

Because the proofs for these claims are straightforward, though somewhat tedious, we leave their verification to the interested reader.

3.2. Measuring Solution Quality in a Particular Instance

Consider now a *particular instance* of the general covering problem on which the k-stage algorithm is applied. A pseudo-greedy implementation of the k-stage algorithm, while guaranteeing a β-approximate optimal solution at each stage, will actually deliver a β_l-approximate solution at stage l, where $\beta_l \geq \beta$, during the particular instance. Similarly, we may only be able to employ a heuristic at stage l for which we can provide no *a priori* performance guarantee, but which turns out to return a subset within a factor of β_l of optimal at stage l. We would like to be able to say something about the quality of solution delivered by the k-stage covering algorithm as a function of the *actual* solution qualities, $\beta_1, \beta_2, \ldots, \beta_k$, found at stages 1, 2, $\ldots$, k, respectively.

We state, without proof, the following lemmas that generalize our earlier ones, and can be proved in a similar manner:

LEMMA 3: $a_l \geq \beta_l\,[c(k) - a(l-1)]/k$, for $l = 1, 2, \ldots, k$.

LEMMA 4: $a(l) \geq (1 - \prod_{i=1}^{l} (1 - \beta_i/k))\, c(k)$, for $l = 1, 2, \ldots, k$.

Let $\vec{\beta} = (\beta_1, \beta_2, \ldots, \beta_k)$, and $\beta_{\text{ave}} = (1/k) \sum_{l=1}^{k} \beta_l$. The main result of this subsection, which expresses the quality of the solution returned in terms of $\vec{\beta}$, follows from Lemma 4:

THEOREM 2: $a(k) \geq \alpha_k(\vec{\beta})c(k)$, where $\alpha_k(\vec{\beta}) = 1 - \prod_{l=1}^{k} (1 - \beta_l/k) > 1 - 1/e^{\beta_{\text{ave}}}$.

PROOF: From Lemma 3.2,

$$a(k) \geq \left(1 - \prod_{l=1}^{k} \left(1 - \frac{\beta_l}{k}\right)\right) c(k) = \alpha_k(\vec{\beta})c(k).$$

Now, because the algebraic mean does not exceed the geometric mean,

$$\sqrt{\prod_{l=1}^{k} \left(1 - \frac{\beta_l}{k}\right)} \leq \frac{1}{k} \sum_{l=1}^{k} \left(1 - \frac{\beta_l}{k}\right) = 1 - \frac{\beta_{\text{ave}}}{k}.$$

Thus,

$$\alpha_k(\vec{\beta}) = 1 - \prod_{l=1}^{k} \left(1 - \frac{\beta_l}{k}\right) \geq 1 - \left(1 - \frac{\beta_{\text{ave}}}{k}\right)^k = \alpha_k(\beta_{\text{ave}}) > 1 - \frac{1}{e^{\beta_{\text{ave}}}}. \qquad \square$$

The above theorem reduces to Theorem 1 when, for all l, $\beta_l = \beta_{\text{ave}}$.

3.3. Finding a $(1 - \epsilon)$-Approximate Cover

Suppose that $C(k)$ is an optimal solution to a general covering problem. We have determined the quality of solution obtained using the greedy or pseudo-greedy implementations of the algorithm of Figure 1, under the assumption that k subsets are selected. Suppose, however, that one is permitted to select more than k subsets, with the goal of covering a group of elements of weight at least $(1 - \epsilon)\, c(k)$.

Let k_ϵ be the number of stages required by the k-stage algorithm to achieve coverage at least $(1 - \epsilon)$ times that of $C(k)$, assuming a pseudo-greedy implementation with parameter β.

THEOREM 3: $k_\epsilon \leq \lceil \log_{1-\beta/k} \epsilon \rceil \leq \lceil (k)/\beta \ln (1/\epsilon) \rceil$.

PROOF: Observe that Lemmas 1 and 2 also hold for $l > k$. By substituting $l \geq \log_{1-\beta/k} \epsilon$ in Lemma 2, we find that $A(l) \geq (1 - \epsilon)c(k)$. It follows that $k_\epsilon \leq \lceil \log_{1-\beta/k} \epsilon \rceil$. Now,

$$\left(1 - \frac{\beta}{k}\right)^k < e^{-\beta} \Rightarrow \left(1 - \frac{\beta}{k}\right)^{-k/\beta \ln \epsilon} > \epsilon \Rightarrow \frac{k}{\beta} \ln \frac{1}{\epsilon} > \log_{1-\beta/k} \epsilon. \qquad \square$$

So, the pseudo-greedy approach guarantees coverage within $(1 - \epsilon)$ of that optimally achieved by k subsets by using at most $O((1/\beta) \log (1/\epsilon))$ times as many subsets.

In many applications, including some in the previous section, the goal may be to cover *all* the elements in U with the fewest number of subsets. Indeed, the natural problem may be to find the smallest k, k^*, such that all elements in U can be covered with k^* subsets. Using the above theorem, we can get a bound on how many stages, using a greedy-like heuristic, are required to cover the entire set U.

COROLLARY 2: Consider an instance of the general covering problem, and let k^* be the optimal number of subsets required to cover all of the elements of the universal set U. Then, using a pseudo-greedy implementation of the generic k-stage covering algorithm, with parameter β, all of the elements of U are covered in k stages, where k/k^* is $O(1/\beta) \log |U|)$.

PROOF: Because we wish to cover all of the elements in U, the weights of the individual elements are unimportant. So, assume each element has unit weight so that the weight of U is $|U|$. Thus, our pseudo-greedy implementation of the k-stage algorithm will select at each stage, within a factor β, that subset that has the maximum number of elements in it. By setting $\epsilon < 1/|U|$ in Theorem 3, we are guaranteed to cover elements with weight exceeding $|U| - 1$, which implies that complete coverage is achieved. Thus, entire coverage is guaranteed to be achieved within k stages, where $k \leq \lceil (k^*/\beta) \ln |U| \rceil$. $\square$

The above corollary implies that the pseudo-greedy approach, for fixed β, covers the entire set U with a number of subsets that is within a factor logarithmic in $|U|$ of the optimal covering. Johnson [17] has actually shown for the *Set Cover* problem, where $\beta = 1$, that the greedy algorithm achieves total coverage with a number of sets that is within a factor logarithmic in $|R_{\max}|$ of the optimal covering, where $R_{\max}$ is the maximum cardinality

subset in $\mathcal{R}$. Furthermore, Chvàtal [5], through an innovative use of duality theory as an analysis tool, gave a similar result for the weighted *Set Cover* problem by using a slightly different greedy implementation of the generic k-stage algorithm. In the weighted *Set Cover* problem, each subset in $\mathcal{R}$ has a weight and the objective is to achieve total coverage with a collection of subsets of minimum total weight. Chvátal's implementation of the generic k-stage algorithm selects that subset in $\mathcal{R}$ that maximizes the ratio of additional coverage achieved to the weight of the subset selected. For additional discussion of approximation algorithms for covering problems the reader is referred to [15].

We conclude this subsection with one further observation. By noting that $c(k) = c(k^*)$ for $k \geq k^*$, the following strengthened versions of Lemmas 1 and 2, whose proofs are left to the interested reader, can be established:

LEMMA 5: $a_l \geq \beta \dfrac{[c(k) - a(l-1)]}{\min\{k, k^*\}}$.

LEMMA 6: $a(l) \geq (1 - (1 - \dfrac{\beta}{\min\{k, k^*\}})^l) c(k)$.

4. SUMMARY

We have provided an analysis of the quality of solution of a greedy algorithm for the general problem of covering the maximum weight set of elements of a universal set via k structurally constrained subsets. As we have shown, many optimization problems fall into this general family of covering problems, and the analysis is therefore applicable to any particular covering problem that employs a strategy of making step-by-step greedy selections. As indicated, performing each greedy step may itself by NP-hard. Our analysis provides a performance bound that is a function of β, a measure of the quality of the pseudo-greedy solution found at each stage; when a greedy selection can be found at each stage, $\beta = 1$.

A nice feature about our k-stage greedy approach is that it provides a sequence of solutions, $A(1), A(2), \ldots, A(k)$ involving $1, 2, \ldots, k$ sets, respectively. Each $A(i)$ is guaranteed to achieve coverage within a factor of $\alpha_i(\beta)$ of the optimal using i sets, for $i = 1, 2, \ldots, k$.

Finally, we point out one direction of further study which is to introduce a limited budget: Here the goal is of covering a maximum weight subset of elements of U, given a bound k on the number of subsets to be selected. Similarly, one can imagine a weighted version in which each subset has an associated cost. The objective, as before, is to cover a maximum weight subset of elements of U, but under the constraint that the total cost of subsets used in the covering does not exceed a ''budget'' B.

REFERENCES

[1] Baker, B.S., ''Approximation Algorithms for NP-Complete Problems on Planar Graphs,'' *Journal of the ACM,* **41**(1), 153–180 (1994).
[2] Bussieck, M., ''The Minimal Cut Cover of a Graph,'' unpublished manuscript, March 1994.
[3] Barahona, F., Weintraub, A., and Epstein, R., ''Habitat Dispersion in Forest Planning and the Stable Set Problem,'' *Operations Research,* **40,** 14–21 (1992).
[4] Cornuejols, G., Fisher, M.L., and Nemhauser, G.L., ''Location of Bank Accounts to Optimize

Float: An Analytic Study of Exact and Approximate Algorithms,'' *Management Science,* **23**(8), 789–810 (1977).
[5] Chvátal, V., ''A Greedy Heuristic for the Set-Covering Problem,'' *Mathematics of Operations Research,* **4**(3), 233–235 (1979).
[6] Cong, J., and Liu, C.L., ''On the *k*-Layer Planar Subset and Via Minimization Problems,'' in *Proceedings of the European Design Automation Conference,* 1990, pp. 459–463.
[7] Dasgupta, B., Janardan, R., and Sherwani, N., ''On the Greedy Algorithm for a Covering Problem,'' unpublished manuscript, February 1993.
[8] Goldschmidt, O., Hochbaum, D.S., Hurken, C., and Yu, G., ''Approximation Algorithms for the *k*-Clique Covering Problem,'' *SIAM Journal of Discrete Mathematics,* **9**(3), 492–509 (1996).
[9] Garey, M.R., and Johnson, D.S., *Computers and Intractability: A Guide to the Theory of NP-Completeness,* Freeman, San Francisco, 1979.
[10] Goemans, M.X., private communication, 1995.
[11] Goemans, M.X., and Williamson, D.P., ''Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming,'' *Journal of the ACM,* **42**(6), 1115–1145 (1995).
[12] Hall, L.A., ''Approximation Algorithms for Scheduling,'' in D.S. Hochbaum (Ed.), *Approximation Algorithms for NP-Hard Problems,* PWS, Boston, 1996, pp. 1–45.
[13] Hochbaum, D.S., and Maass, W., ''Approximation Schemes for Covering and Packing Problems in Image Processing and VLSI,'' *Journal of the Association for Computing Machinery,* **32**(1), 130–136 (1985).
[14] Hochbaum, D.S., ''Efficient Bounds for the Stable Set, Vertex Cover and Set Packing Problems,'' *Discrete Applied Mathematics,* **6,** 243–254 (1983).
[15] Hochbaum, D.S., ''Approximation Covering and Packing Problems: Set Cover, Vertex Cover, Independent Set, and Related Problems,'' in D.S. Hochbaum (Ed.), *Approximation Algorithms for NP-Hard Problems.* PWS, Boston, 1996, pp. 94–143.
[16] Ho, J.M., Sarrafzadeh, M., Vijayan, G., and Wong, C.K., ''Layer Assignment for Multichip Modules,'' *IEEE Transactions on Computer-Aided Design,* **9,** 1272–1277 (1990).
[17] Johnson, D.S., ''Approximation Algorithms for Combinatorial Problems,'' *Journal of Computer and System Sciences,* **9,** 256–278 (1974).
[18] Loulou, R., ''Minimal Cut Cover of a Graph with an Application to the Testing of Electronic Boards,'' *Operations Research Letters,* **12**(5), 301–306 (1992).
[19] Lipton, R.J., and Tarjan, R.E., ''Applications of a Planar Separator Theorem,'' *SIAM Journal of Computing,* **9**(3), 615–627 (1980).
[20] Linial, N., and Vazirani, U., ''Graph Products and Chromatic Numbers,'' in *30th Annual Symposium on Foundations of Computer Science,* 1989, pp. 124–128.
[21] Megiddo, N., Zemel, E., and Hakimi, S.L., ''The Maximum Coverage Location Problem,'' *SIAM Journal of Algebraic and Discrete Methods,* **4**(2), 253–261 (1983).
[22] Supowit, K., ''Finding a Maximum Planar Subset of a Set of Nets in a Channel,'' *IEEE Transactions on Computer-Aided Design,* **6,** 93–94 (1987).