RUTCOR
RESEARCH
REPORT

A GREEDY HEURISTIC FOR A GENERALIZED SET COVERING PROBLEM

Peter Chen[a] Guoli Ding[b]

RRR 16-2008,

RUTCOR
Rutgers Center for
Operations Research
Rutgers University
640 Bartholomew Road
Piscataway, New Jersey
08854-8003
Telephone: 732-445-3804
Telefax: 732-445-5472
Email: rrr@rutcor.rutgers.edu
http://rutcor.rutgers.edu/~rrr

[a]Department of Computer Science, Louisiana State University, Baton Rouge, Louisiana, 70803, USA (pchen@lsu.edu)
[b]Department of Mathematics, Louisiana State University, Baton Rouge, Louisiana, 70803, USA (ding@math.lsu.edu)

Rutcor Research Report
RRR 16-2008,

A greedy heuristic for a generalized set covering problem

Peter Chen Guoli Ding

Abstract. The classical weighted set covering problem is generalized simultaneously in three directions. First, each numerical weight is replaced by a weighted set, which we call *cost*. Second, each element in the ground set is assigned a numerical weight. Third, the concept of a cover is relaxed to a partial cover that only needs to cover some percentage of the ground set, instead of the whole ground set. The last two generalizations have been studied in the literature, while the first is new. We propose a greedy algorithm to approximate this generalized problem and we establish an upper bound on the ratio of the greedy solution over the optimal solution. This bound is independent of the cost function, and it depends only on the total weight of the ground set. We prove that our bound is the best possible.

Keywords: Set cover, Greedy algorithm.

Acknowledgements: The authors are indebted to Robert Lax and Nigel Gwee for their valuable comments and suggestions. This research is partially supported by NSF grants DMS-0556091 and ITR-0326387.

1 Introduction

The purpose of this paper is to introduce a generalized set covering problem (GSCP), and to propose, in Section 2, a greedy algorithm to approximate it. We prove, in Theorem 1, that our solution is not too far away from the optimal solution. We also prove, in Theorem 2, that the bound given in Theorem 1 is the best possible.

For clarity, we first have some definitions. If $\mathcal{X}$ is a finite collection of sets, then $\overline{\mathcal{X}}$ is the union of all members of $\mathcal{X}$. If f is a function from a set X to R_+, the set of nonnegative reals, then, for any finite subset X' of X, $f(X')$ is defined to be the sum of $f(x)$, over all x in X'.

Let S be a finite set and let $\mathcal{S} = \{S_1, S_2, \ldots, S_n\}$, where each S_i is a subset of S. We call $\mathcal{A} \subseteq \mathcal{S}$ a *cover* of $\mathcal{S}$ if $\overline{\mathcal{A}} = \overline{\mathcal{S}}$. The classical *set covering problem* (SCP) is to find a cover with a minimum cardinality. In applications, there is usually a *weight* function w from $\mathcal{S}$ to R_+. In such a situation, the *total weight* of a cover $\mathcal{A}$ is defined to be $w(\mathcal{A})$. The *weighted SCP* (WSCP) is to find a cover with a minimum total weight. Clearly, WSCP is a generalization of SCP, as WSCP is SCP when $w(S_i) = 1$, for all i.

In this paper, we study a generalization of WSCP, which arose from the authors' work on profiling. We will relax WSCP in three directions. First, we replace each numerical weight $w(S_i)$ with a weighted set. Second, we give every element in S a numerical weight. Third, we only require $\mathcal{A}$ to cover a portion of $\overline{\mathcal{S}}$, instead of the entire $\overline{\mathcal{S}}$.

Let S and $\mathcal{S}$ be as before. Let d be a function from S to R_+ and let $\lambda \in [0, 1]$. Then $\mathcal{A} \subseteq \mathcal{S}$ is called a *λ-d-cover* of $\mathcal{S}$ if $d(\overline{\mathcal{A}}) \geq \lambda d(\overline{\mathcal{S}})$. Notice that, when $d(x)$ is positive for all $x \in S$, then $\mathcal{A}$ is a cover if and only if it is a 1-d-cover. Therefore, "λ-d-cover" is a generalization of "cover".

Let W be a finite set, c be a function from W to R_+, and $\mathcal{W} = \{W_1, W_2, \ldots, W_n\}$, where each W_i is a subset of W. We consider each W_i as the *weight* of S_i. For any $\mathcal{A} \subseteq \mathcal{S}$, we define $\mathcal{W}(\mathcal{A}) = \bigcup\{W_i : S_i \in \mathcal{A}\}$ and we call $c(\mathcal{W}(\mathcal{A}))$ the *cost* of $\mathcal{A}$. In particular, if $W = \{1, 2, ..., n\}$, and if for each i we have $W_i = \{i\}$ and $c(i) = w(S_i)$, then it is easy to see that $c(\mathcal{W}(\mathcal{A}))$ is exactly $w(\mathcal{A})$. Therefore, "cost" is a generalization of "total weight".

Generalized SCP (GSCP): *For any given $(S, W, \mathcal{S}, \mathcal{W}, d, c, \lambda)$, find a λ-d-cover of $\mathcal{S}$ with the minimum cost.*

In case $W = \{1, 2, ..., n\}$, $\mathcal{W} = \{\{1\}, \{2\}, ..., \{n\}\}$, and $d(x) = 1$, for all $x \in S$, our GSCP is known as *partial set cover problem* (PSCP) [10], which has the objective of finding $\mathcal{A} \subseteq \mathcal{S}$ with $|\overline{\mathcal{A}}| \geq \lambda|\overline{\mathcal{S}}|$ and such that $w(\mathcal{A})$ is minimized.

GSCP is also related to the *submodular set cover problem* (SSCP) [14], which we briefly describe below. Let U be a finite set and let f be a function from 2^U to the set of nonnegative integers such that: (i) $f(X) \leq f(Y)$ for all $X \subseteq Y \subseteq U$, and (ii) $f(X) + f(Y) \geq f(X \cap Y) + f(X \cup Y)$ for all $X, Y \subseteq U$. Let w be a function from U to R_+. The objective of SSCP is to find $A \subseteq U$ with $f(A) = f(U)$ and such that $w(A) = \sum_{a \in A} w(a)$ is minimized. It is not difficult to verify (see [4]) that SSCP is a special case of GSCP if $U = \mathcal{S}$ and $f(\mathcal{A}) = \min\{\lambda d(\bar{\mathcal{S}}), d(\bar{\mathcal{A}})\}$, for all $\mathcal{A} \subseteq \mathcal{S}$, which is exactly the case of GSCP with $W = \{1, 2, ..., n\}$ and $\mathcal{W} = \{\{1\}, \{2\}, ..., \{n\}\}$. Since the objective function of SSCP is linear

while the objective function of GSCP is not, GSCP in general is not a special case of SSCP. On the other hand, since the constraint in SSCP is a general submodular function while the constraint in GSCP is a special submodular function (the function $f(\mathcal{A})$ defined above, see [4]), SSCP in general is not a special case of GSCP either. Therefore, GSCP and SSCP are incomparable, as illustrated in Figure 1.

Figure 1: GSCP and SSCP are different generalizations of PSCP.

There are other related problems. The *minimum k-set cover problem* [5] seeks $\mathcal{A} \subseteq \mathcal{S}$ with $|\bar{\mathcal{A}}| \geq k$ and such that $|\mathcal{A}|$ is as small as possible. The *maximum k-set cover problem* [6, 7] seeks $\mathcal{A} \subseteq \mathcal{S}$ with $|\mathcal{A}| \leq k$ and such that $\overline{\mathcal{A}}$ is as large as possible. Clearly, one may also generalize these problems in the same way as we generalize SCP. In fact, we have considered these generalizations and obtained similar results. These results will appear elsewhere.

It is well-known [9, 13] that SCP is NP-hard. As we discussed above, GSCP contains WSCP as a special case, which in turn contains SCP as a special case. Therefore, GSCP is also NP-hard. In recent years, more hardness results have been proved for SCP. Raz and Safra [12], and Arora and Sudan [1] proved that, for some $c > 0$, there is no polynomial time algorithm that can approximate SCP to a factor of $c \ln |S|$, unless P=NP. Moreover, Feige [3] proved that, for all $\epsilon > 0$, there is no polynomial time algorithm that can approximate SCP to a factor of $(1 - \epsilon) \ln |S|$, unless $NP \subseteq DTIME(n^{O(\ln \ln n)})$ (this assumption is stronger than $P \neq NP$, but it is also widely believed to be true). Clearly, these negative results hold for the more general problems WSCP and GSCP as well.

On the positive side, the best known polynomial time algorithm that approximates WSCP is a greedy algorithm proposed by Chvatal [2], which generalizes earlier results of Johnson [8] and Lovasz [11]. When comparing the solution found by this algorithm with the optimal solution, it is proved that the ratio is at most $1 + \ln s$, where $s = \max\{|S_i| : i = 1, 2, ..., n\}$.

In the next section, we present a polynomial time approximation algorithm, called **GSCA**, for GSCP. This is a modification of Chvatal's algorithm.

Theorem 1. *For any instance $I = (S, W, \mathcal{S}, \mathcal{W}, d, c, \lambda)$ of GSCP, if $Opt(I)$ is the minimum cost of I and $GSCA(I)$ is the cost obtained by* **GSCA***, then*

$$GSCA(I) \leq \begin{cases} (1 + \lambda d(\overline{\mathcal{S}})/d_{min}) \cdot Opt(I) & \text{if } \lambda \in [0, 1) \\ (d(\overline{\mathcal{S}})/d_{min}) \cdot Opt(I) & \text{if } \lambda = 1 \end{cases}$$

where $d_{min} = \min\{d(x) : x \in \overline{\mathcal{S}}, d(x) > 0\}$, and $d(\overline{\mathcal{S}})/d_{min}$ is defined to be 1 if $d(x) = 0$ for all $x \in \overline{\mathcal{S}}$.

In particular, when $d(x) = 1$, for all $x \in S$, it is obvious that Theorem 1 can be simplified as

$$GSCA(I) \leq \begin{cases} (1 + \lambda |\overline{\mathcal{S}}|) \cdot Opt(I) & \text{if } \lambda \in [0, 1), \\ |\overline{\mathcal{S}}| \cdot Opt(I) & \text{if } \lambda = 1. \end{cases}$$

Remark 1. Notice that, as in Chvatal's result, the ratio $GSCA(I)/Opt(I)$ is bounded by a function that is independent of $\mathcal{S}$, W, $\mathcal{W}$, and c. However, unlike Chvatal's results, the bound is a linear function, instead of a logarithmic function, of $d(\overline{\mathcal{S}})$.

Our next result shows that, for **GSCA**, this bound is the best possible.

Theorem 2. *If* $\lambda \in [0,1)$, *then* $\sup_I \dfrac{GSCA(I)}{(1+\lambda d(\overline{\mathcal{S}})/d_{min}) \cdot Opt(I)} = 1$;

If $\lambda = 1$, *then* $\sup_I \dfrac{GSCA(I)}{(d(\overline{\mathcal{S}})/d_{min}) \cdot Opt(I)} = 1.$

Remark 2. We will see from our proof of Theorem 2 that the supremum is already 1 even when the instances are restricted to those with bounded $\mathcal{S}$ and $d(x) = 1$, for all $x \in S$.

It is interesting to consider the Boolean formulation of **GSCA**. Let $S = \{s_1, s_2, ..., s_p\}$. Then SCP can be formulated as follows, where $x_1, x_2, ..., x_n$ are Boolean variables.

$$\text{Minimize} \left\{ \sum_{i=1}^{n} x_i \ : \ \sum_{j=1}^{p} \prod_{S_i \ni s_j} \bar{x}_i = 0 \right\}.$$

Let $W = \{w_1, w_2, ..., w_q\}$. Then **GSCA** can be expressed as follows.

$$\text{Minimize} \left\{ \sum_{j=1}^{q} c_j \bigvee_{W_i \ni w_j} x_i \ : \ \sum_{j=1}^{p} d_j \bigvee_{S_i \ni s_j} x_i \geq \lambda \sum_{j=1}^{p} d_j \right\}. \tag{1}$$

Using formula $\bigvee_{u \in U} u = 1 - \prod_{u \in U} \bar{u}$ and substitution $y_i = \bar{x}_i$, we can equivalently rewrite (1) as

$$\text{Maximize} \left\{ \sum_{j=1}^{q} c_j \prod_{W_i \ni w_j} y_i \ : \ \sum_{j=1}^{p} d_j \prod_{S_i \ni s_j} y_i \leq b \right\} \tag{2}$$

where $b = (1-\lambda)d(S)$. It is interesting to see that **GSCA** is equivalent to a very general Boolean optimization problem (2). It is also interesting to see that a greedy algorithm can produce a reasonable solution.

In the next section, we describe algorithm **GSCA**. Then, in section 3, we prove the two theorems listed above. We also give an example that shows, in Theorem 1, $1 + \lambda d(\overline{\mathcal{S}})/d_{min}$ can not be replaced with $\lambda d(\overline{\mathcal{S}})/d_{min}$ for general λ. Finally, in the last section, we discuss some possible improvements of **GSCA**.

2 Algorithm GSCA

The basic idea for Chvatal's algorithm is to bring, at each iteration, the least expensive set into the proposed cover. We will still use the same idea. However, since our weight functions are much more complicated, our update procedure, at each iteration, is more complicated.

The algorithm starts with $\mathcal{A} = \varnothing$, which, in the end of the algorithm, will be a λ-d-cover of $\mathcal{S}$. At each iteration, one new set from $\mathcal{S}$ is chosen and added to $\mathcal{A}$, until $\mathcal{A}$ becomes a λ-d-cover of $\mathcal{S}$. Therefore, at the beginning of, say, the k-th iteration, $\mathcal{A}$ contains $k-1$ members. Without loss of generality, let us assume that S_1, S_2, ..., S_{k-1} are these sets. Let

$$S' = S_1 \cup S_2 \cup ... \cup S_{k-1} \quad \text{and} \quad W' = W_1 \cup W_2 \cup ... \cup W_{k-1}.$$

Since we are only interested in covering the remaining elements of S, we define

$$\mathcal{S}' = \{S'_i = S_i - S' : i = k, k+1, ..., n\}.$$

Because of the way we compute cost, we also define

$$\mathcal{W}' = \{W'_i = W_i - W' : i = k, k+1, ..., n\}.$$

We point out that $d(S') = d(\overline{\mathcal{A}}) < \lambda d(\overline{\mathcal{S}})$, because otherwise, $\mathcal{A}$ would already be a λ-d-cover and we would have stopped.

If we add S_i to $\mathcal{A}$, then, with an extra cost of $c(W'_i)$, we can cover an extra set S'_i of elements. It follows that, in general, the *average cost* for covering elements in S'_i is

$$ac(S'_i) = c(W'_i)/d(S'_i),$$

where the value is defined to be ∞ if $d(S'_i) = 0$. An exception to this calculation happens when

$$d(S') + d(S'_i) = d(S' \cup S'_i) > \lambda d(\overline{\mathcal{S}}).$$

In this case, if we add S_i to $\mathcal{A}$, we will cover more than we are required to. Since we should not pay any extra for any extra coverage, the actual average cost for covering elements in S'_i should be

$$ac(S'_i) = c(W'_i)/(\lambda d(\overline{\mathcal{S}}) - d(S')),$$

instead of the smaller value $c(W'_i)/d(S'_i)$. We remark that the above denominator $\lambda d(\overline{\mathcal{S}}) - d(S')$ is not zero. In fact, it is positive, as $\mathcal{A}$ is not a λ-d-cover yet.

Now the key ingredient of our algorithm can be stated as follows: At each iteration, we select i to minimize $ac(S'_i)$, and then add S_i to $\mathcal{A}$.

Notice that elements in $S - \overline{\mathcal{S}}$ and elements in $W - \overline{\mathcal{W}}$ do not play any role in our problem. Therefore, in our algorithm, we will consider $(\mathcal{S}, \mathcal{W}, d, c, \lambda)$ as the input, not $(S, W, \mathcal{S}, \mathcal{W}, d, c, \lambda)$.

The General Set Covering Algorithm (GSCA).

GSCA $(\mathcal{S}, \mathcal{W}, d, c, \lambda)$:

Step 0. Initialize.

Set $\mathcal{A} \leftarrow \varnothing$.
Set $S'_i \leftarrow S_i$ and $W'_i \leftarrow W_i$, for $i = 1, 2, ..., |\mathcal{S}|$.

Step 1. Select a new set.

If $d(\overline{\mathcal{A}}) \geq \lambda d(\overline{\mathcal{S}})$, then, output $\mathcal{A}$, which is a λ-d-cover of $\mathcal{S}$, and stop.
Otherwise, for each $S_j \in \mathcal{S} - \mathcal{A}$, compute

$$ac(j) = \begin{cases} \infty & \text{if } d(S'_j) = 0, \\ c(W'_j)/d(S'_j) & \text{if } d(S'_j) \neq 0 \text{ and } d(\overline{\mathcal{A}} \cup S'_j) \leq \lambda d(\overline{\mathcal{S}}), \\ c(W'_j)/(\lambda d(\overline{\mathcal{S}}) - d(\overline{\mathcal{A}})) & \text{if } d(S'_j) \neq 0 \text{ and } d(\overline{\mathcal{A}} \cup S'_j) > \lambda d(\overline{\mathcal{S}}). \end{cases}$$

Then find an index i_{min} with

$$ac(i_{min}) = \min\{ac(j) : S_j \in \mathcal{S} - \mathcal{A}\}$$

and proceed to Step 2.

Step 2. Update.

Set $\mathcal{A} \leftarrow \mathcal{A} \cup \{S_{i_{min}}\}$.
Set $S'_k \leftarrow S'_k - S_{i_{min}}$ and $W'_k \leftarrow W'_k - W_{i_{min}}$, for every $S_k \in \mathcal{S} - \mathcal{A}$.
Then, return to Step 1.

It is worth pointing out that, if $\lambda = 1$, then the computation on $ac(j)$ can be simplified, as the last alternative can be eliminated.

Proposition. GSCA *runs in polynomial time and its output is a λ-d-cover of $\mathcal{S}$.*

Proof. At the beginning of each iteration, if the current $\mathcal{A}$ is not a λ-d-cover yet, then $\lambda d(\overline{\mathcal{S}}) - d(\overline{\mathcal{A}}) > 0$. It follows that there is at least one $S_j \in \mathcal{S} - \mathcal{A}$ for which $d(S'_j) > 0$. Consequently, a new $\mathcal{A}$ will be generated at the end of this iteration, and the new $\mathcal{A}$ will contain one more member than the old $\mathcal{A}$. Therefore, **GSCA** terminates only when $\mathcal{A}$ becomes a λ-d-cover, and does so after at most $|\mathcal{S}|$ iterations. It is obvious that each iteration takes only polynomial time, so the entire algorithm runs in polynomial time. Since the algorithm terminates eventually and it does not terminate while $d(\overline{\mathcal{A}}) < \lambda d(\overline{\mathcal{S}})$, we conclude that the output must be a λ-d-cover of $\mathcal{S}$. ∎

3 Proofs of the theorems

In this section, we prove the two theorems stated in the introduction. We first prove Theorem 1.

Proof. We prove the theorem by proving a sequence of claims. As before, let $n = |\mathcal{S}|$.

Claim 1. We may assume that $d(S_j) > 0$, for all $S_j \in \mathcal{S}$.

Suppose $\mathcal{S}$ has a member S_j with $d(S_j) = 0$. Then we may assume that S_j does not appear in the optimal solution. On the other hand, when **GSCA** is applied, we have $ac(j) = \infty$ throughout the whole process. Notice that, at each iteration, when a new index i_{min} is selected, we must have $ac(i_{min}) < \infty$, as the current $\mathcal{A}$ is not a λ-d-cover yet. Therefore, j will never be selected and thus S_j will not appear in the output of **GSCA**. In conclusion, the theorem holds for $(S, W, \mathcal{S}, \mathcal{W}, d, c, \lambda)$ if and only if it holds for $(S, W, \mathcal{S} - \{S_j\}, \mathcal{W}, d, c, \lambda)$, which means we may assume that $d(S_j) > 0$, for all $S_j \in \mathcal{S}$.

Claim 2. We may assume that $d(x) > 0$ holds for at least one $x \in \overline{\mathcal{S}}$.

If $d(x) = 0$, for all $x \in \overline{\mathcal{S}}$, then $\mathcal{A} = \varnothing$ is a λ-d cover of $\mathcal{S}$. Moreover, it is easy to see that $\mathcal{A}$ is both an optimal solution and a greedy solution. That is, $Opt(I) = GSCA(I) = 0$. Therefore, the theorem holds in this case, which implies that we may assume, for at least one $x \in \overline{\mathcal{S}}$, that $d(x) > 0$.

Claim 3. We may assume that $d_{min} = 1$.

By Claim 2, we have $d_{min} > 0$. Let $d'(x) = d(x)/d_{min}$, for all $x \in S$. Then $d'_{min} = 1$. By the definition of a λ-d-cover, it is clear that $\mathcal{A} \subseteq \mathcal{S}$ is a λ-d'-cover of $\mathcal{S}$ if and only if it is a λ-d-cover of $\mathcal{S}$. In addition, it is also easy to verify that $\mathcal{A} \subseteq \mathcal{S}$ is the output of $\mathbf{GSCA}(\mathcal{S}, \mathcal{W}, d', c, \lambda)$ if and only if it is the output of $\mathbf{GSCA}(\mathcal{S}, \mathcal{W}, d, c, \lambda)$. Therefore, Theorem 1 holds for $(\mathcal{S}, \mathcal{W}, d, c, \lambda)$ if and only if it holds for $(\mathcal{S}, \mathcal{W}, d', c, \lambda)$. This equivalence proves Claim 3.

Claim 4. We may assume that $\lambda d(\overline{\mathcal{S}}) \geq 1$.

We only need to show that the theorem holds when $\lambda d(\overline{\mathcal{S}}) < 1$. Suppose we are in such a situation. By claims 1 and 3, every member of $\mathcal{S}$ forms a λ-d-cover of $\mathcal{S}$. Then, we deduce from Claim 2 that $\mathcal{S} \neq \varnothing$, and so, $\mathcal{S}$ has a member, say S_1, such that $c(W_1) \leq c(W_j)$, for all j. Consequently, $\{S_1\}$ is an optimal solution with $Opt(I) = c(W_1)$. On the other hand, when **GSCA** is applied to this instance, it finishes in only one iteration. At the very beginning, we have $\mathcal{A} = \varnothing$. By Claim 1 and Claim 3,

$$d(\overline{\mathcal{A}} \cup S_j) = d(S_j) \geq 1 > \lambda d(\overline{\mathcal{S}}), \quad j = 1, 2, ..., n,$$

and it follows that

$$ac(j) = c(W_j)/(\lambda d(\overline{\mathcal{S}})), \quad j = 1, 2, ..., n.$$

By the definition of S_1, **GSCA** will pick $i_{min} = 1$, since $ac(1) \leq ac(j)$, for all j. It means that $\{S_1\}$ is the output of **GSCA**. Therefore, $Opt(I) = GSCA(I)$ and thus the theorem holds. Claim 4 is proved.

Claim 5. We may assume that the output $\mathcal{A}$ of **GSCA** has at least two members.

It is enough for us to show that the theorem holds when $|\mathcal{A}| \leq 1$. Let $\mathcal{K} \subseteq \mathcal{S}$ be an optimal solution. From Claim 2 we deduce that neither $\mathcal{A}$ nor $\mathcal{K}$ is empty. Without loss of generality, let S_1 be the only member of $\mathcal{A}$ and let S_k be a member of $\mathcal{K}$, where k could be

1. Then $d(S_1) \geq \lambda d(\overline{\mathcal{S}})$, as $\mathcal{A} = \{S_1\}$ is the output of **GSCA**, and thus must be a λ-d-cover of $\mathcal{S}$. When S_1 was selected, in the first iteration of **GSCA**, we must have

$$c(W_1)/(\lambda d(\overline{\mathcal{S}})) = ac(1) \leq ac(k).$$

Since this is the first iteration of **GSCA**, we deduce from claims 1, 3, and 4, that

$$ac(k) \leq c(W_k) \leq c(\mathcal{W}(\mathcal{K})) = Opt(I).$$

Then, combining the last two inequalities results in

$$GSCA(I) = c(W_1) \leq \lambda d(\overline{\mathcal{S}}) Opt(I),$$

and thus the theorem holds, which proves Claim 5.

Next, we consider the general case. Let

$$\sigma = \begin{cases} 1 + \lambda d(\overline{\mathcal{S}}) & \text{if } \lambda \in [0, 1), \\ d(\overline{\mathcal{S}}) & \text{if } \lambda = 1. \end{cases}$$

Suppose the theorem is false. Then, by Claim 3, there exists an instance $(\mathcal{S}, \mathcal{W}, d, c, \lambda)$, for which

$$c(\mathcal{W}(\mathcal{A})) > \sigma \cdot c(\mathcal{W}(\mathcal{K})), \tag{1}$$

where $\mathcal{A}$ is the output of **GSCA** and $\mathcal{K}$ is a λ-d-cover of $\mathcal{S}$. We choose such an instance with $|\overline{\mathcal{S}}|$ as small as possible. We will deduce a contradiction and that will prove the validity of the theorem.

By Claim 2, $\mathcal{A} \neq \varnothing$. Without loss of generality, we may assume that $S_1 \in \mathcal{S}$ is the set selected by **GSCA** in the first iteration. Then, by Claim 5, we know that $\mathcal{K} \neq \{S_1\}$. Therefore, $\mathcal{K}$ must contain some S_k with $k \neq 1$. Without loss of generality, let $k = 2$. By considering the first iteration of **GSCA**, we have

$$\begin{aligned} c(W_1)/d(S_1) &= ac(1) && \text{(by Claim 5)} \\ &\leq ac(2) && \text{(by the definition of } S_1) \\ &\leq c(W_2) && \text{(by claims 3 and 4)} \\ &\leq c(\mathcal{W}(\mathcal{K})), \end{aligned}$$

which gives us

$$c(W_1) \leq d(S_1) \cdot c(\mathcal{W}(\mathcal{K})). \tag{2}$$

Let us consider the instance $(\mathcal{S}', \mathcal{W}', d, c, \lambda')$, where

$$\begin{aligned} \mathcal{S}' &= \{S_i - S_1 : i = 1, 2, ..., n\}, \\ \mathcal{W}' &= \{W_i - W_1 : i = 1, 2, ..., n\}, \quad \text{and} \\ \lambda' &= (\lambda d(\overline{\mathcal{S}}) - d(S_1))/d(\overline{\mathcal{S}'}). \end{aligned}$$

Then it is clear that

$$S_1 \subseteq \overline{\mathcal{S}} \text{ and } \overline{\mathcal{S}'} = \overline{\mathcal{S}} - S_1. \tag{3}$$

Claim 6. $\lambda' \in [0,1]$. Moreover, $\lambda' = 1$ if and only if $\lambda = 1$.

By Claim 5, S_1 is not the only member of $\mathcal{A}$. It follows that $d(S_1) < \lambda d(\overline{\mathcal{S}})$, and so $\lambda' \geq 0$. On the other hand, by (3), we have

$$\lambda' = \lambda - (1-\lambda)d(S_1)/d(\overline{\mathcal{S}'}) \leq \lambda \leq 1,$$

which proves the first part of Claim 6. Furthermore, the last inequality also implies that second part of Claim 6, so the proof of the claim is complete.

Let

$$\sigma' = \begin{cases} 1 + \lambda' d(\overline{\mathcal{S}'}) & \text{if } \lambda' \in [0,1), \\ d(\overline{\mathcal{S}'}) & \text{if } \lambda' = 1. \end{cases}$$

Then, by (3), the second part of Claim 6, and the definitions of σ and λ', it is easy to verify that

$$\sigma = \sigma' + d(S_1). \tag{4}$$

From our choice of S_1 and the definitions of $\mathcal{S}'$, $\mathcal{W}'$, and λ', it is routine to verify that, when **GSCA** is applied to the new instance, at iteration k, each $ac(j)$ is exactly the same as the corresponding value at iteration $k+1$ when **GSCA** is applied to the original instance. Therefore, when **GSCA** is applied to the new instance, the output is $\mathcal{A}' = \mathcal{A} - \{S_1\}$. On the other hand, it is clear from the definition of λ' that $\mathcal{K}' = \{S_i - S_1 : S_i \in \mathcal{K}\}$ is a λ'-d-cover of $\mathcal{S}'$. From (3), Claim 1, the minimality of $|\overline{\mathcal{S}}|$, and Claim 3 we deduce that

$$c(\mathcal{W}'(\mathcal{A}')) \leq \sigma' \cdot c(\mathcal{W}'(\mathcal{K}')) \tag{5}$$

It is straightforward to verify that

$$W_1 \subseteq \mathcal{W}(\mathcal{A}) \text{ and } \mathcal{W}'(\mathcal{A}') = \mathcal{W}(\mathcal{A}) - W_1 \tag{6}$$

and then

$$\begin{aligned}
&c(\mathcal{W}(\mathcal{A})) - \sigma c(\mathcal{W}(\mathcal{K})) \\
&= c(W_1) + c(\mathcal{W}'(\mathcal{A}')) - \sigma c(\mathcal{W}(\mathcal{K})) && \text{(by (6))} \\
&\leq c(W_1) + \sigma' c(\mathcal{W}'(\mathcal{K}')) - \sigma c(\mathcal{W}(\mathcal{K})) && \text{(by (5))} \\
&= c(W_1) + (\sigma - d(S_1)) c(\mathcal{W}'(\mathcal{K}')) - \sigma c(\mathcal{W}(\mathcal{K})) && \text{(by (4))} \\
&= c(W_1) - \sigma(c(\mathcal{W}(\mathcal{K})) - c(\mathcal{W}'(\mathcal{K}'))) - d(S_1) c(\mathcal{W}'(\mathcal{K}')) \\
&\leq c(W_1) - d(S_1)(c(\mathcal{W}(\mathcal{K})) - c(\mathcal{W}'(\mathcal{K}'))) - d(S_1) c(\mathcal{W}'(\mathcal{K}'))) && \text{(by (4))} \\
&= c(W_1) - d(S_1) c(\mathcal{W}(\mathcal{K})) \\
&\leq 0, && \text{(by (2))}
\end{aligned}$$

contradicting (1). The theorem is proved. ∎

Next, we present an example to show that, in Theorem 1, $1 + \lambda d(\overline{\mathcal{S}})/d_{min}$ can not be replaced by $\lambda d(\overline{\mathcal{S}})/d_{min}$, for $\lambda \in [0, 1)$.

Example 1. Let $n = m + 1$, where $m > 1$, and let $\varepsilon > 0$ be a small real number. Let

$$\begin{aligned}
S &= \{s_1, s_2, ..., s_m\} \\
W &= \{w_0, w_1, w_2, ..., w_m, w_{m+1}\} \\
S_i &= \{s_i\} && \text{for } i = 1, 2, ..., m \\
S_{m+1} &= \{s_1, s_2, ..., s_m\} \\
W_i &= \{w_0, w_i\} && \text{for } i = 1, 2, ..., m \\
W_{m+1} &= \{w_{m+1}\} \\
c(w_0) &= 1 \\
c(w_i) &= \varepsilon && \text{for } i = 1, 2, ..., m \\
c(w_{m+1}) &= m \\
d(s_i) &= 1 && \text{for } i = 1, 2, ..., m \\
\lambda &= (2m - 1)/(2m).
\end{aligned}$$

Since $\lambda d(S) = (2m-1)/2 > m-1 = d(S)-1$, and d is an integral function, every λ-d-cover of $\mathcal{S}$ must be a cover of $\mathcal{S}$. Observe that there are two minimal covers $\mathcal{A}_1 = \{S_1, ..., S_m\}$ with $c(\mathcal{W}(\mathcal{A}_1)) = c(\{w_0, w_1, ..., w_m\}) = 1 + m\varepsilon$ and $\mathcal{A}_2 = \{S_{m+1}\}$ with $c(\mathcal{W}(\mathcal{A}_2)) = c(w_{m+1}) = m$. It follows that $Opt(I) = 1+m\varepsilon$. When **GSCA** is applied to this instance, It is straightforward to verify that $\mathcal{A}_2$ is the output, which means that $GSCA(I) = m$. However, when ε is sufficiently small, we have

$$\lambda d(S) Opt(I) = (1 + m\varepsilon)(m - \frac{1}{2}) < m = GSCA(I),$$

which shows that, in Theorem 1, $1 + \lambda d(\overline{\mathcal{S}})/d_{min}$ can not be replaced by $\lambda d(\overline{\mathcal{S}})/d_{min}$, for $\lambda \in [0, 1)$, not even when $d(x) = 1$, for all $x \in S$.

Next, we prove Theorem 2.

Proof. We consider the two cases $\lambda \in [0, 1)$ and $\lambda = 1$ separately. In each case, we need to find a

sequence of instances, such that the limit of the corresponding expression is 1. In both cases, we will have $d(x) = 1$, for all $x \in S$. We first consider the case $\lambda \in [0, 1)$. Let $n = m + 1$,

where $m > 1$, and let $0 < \varepsilon < 1/m$ be a small real number. Let

$$\begin{aligned}
S &= \{s_1, s_2, ..., s_m\} \\
W &= \{w_0, w_1, w_2, ..., w_m, w_{m+1}\} \\
S_i &= \{s_i\} && \text{for } i = 1, 2, ..., m \\
S_{m+1} &= \{s_1, s_2, ..., s_m\} \\
W_i &= \{w_0, w_i\} && \text{for } i = 1, 2, ..., m \\
W_{m+1} &= \{w_{m+1}\} \\
c(w_0) &= 1 \\
c(w_i) &= \varepsilon && \text{for } i = 1, 2, ..., m \\
c(w_{m+1}) &= m \\
d(s_i) &= 1 && \text{for } i = 1, 2, ..., m \\
\lambda &= (m - 1 + \varepsilon)/m.
\end{aligned}$$

This is a modification of Example 1. As in that example, $\lambda d(S) > m - 1 = d(S) - 1$. Then, with the same argument, we can see that $\mathcal{A}_1 = \{S_1, ..., S_m\}$ is an optimal solution with

$$Opt(I) = c(\mathcal{W}(\mathcal{A}_1)) = c(\{w_0, w_1, ..., w_m\}) = 1 + m\varepsilon.$$

In addition, the output of **GSCA** must be $\mathcal{A}_2 = \{S_{m+1}\}$, which implies that

$$GSCA(I) = c(\mathcal{W}(\mathcal{A}_2)) = c(w_{m+1}) = m.$$

Now from $\lambda d(S) = m - 1 + \varepsilon$, we conclude that, as $\varepsilon \to 0$,

$$\frac{GSCA(I)}{(1 + \lambda d(\overline{\mathcal{S}})/d_{min}) \cdot Opt(I)} = \frac{m}{(m + \varepsilon)(1 + m\varepsilon)} \to 1,$$

which proves the first part of Theorem 2.

Next, we consider the case when $\lambda = 1$. Let $n = 3m$ and let $0 < \varepsilon < 1/(2m)$. Let

$$\begin{aligned}
S &= \{s_1, s_2, ..., s_{2m}\} \\
W &= \{w_0, w_1, w_2, ..., w_{3m}\} \\
S_i &= \{s_i\} && \text{for } i = 1, 2, ..., 2m \\
S_{2m+j} &= \{s_{2j-1}, s_{2j}\} && \text{for } j = 1, 2, ..., m \\
W_i &= \{w_0, w_i\} && \text{for } i = 1, 2, ..., 2m \\
W_{2m+j} &= \{w_{2m+j}\} && \text{for } j = 1, 2, ..., m \\
c(w_0) &= 1 \\
c(w_i) &= \varepsilon && \text{for } i = 1, 2, ..., 2m \\
c(w_{2m+j}) &= 2 && \text{for } j = 1, 2, ..., m \\
d(s_i) &= 1 && \text{for } i = 1, 2, ..., 2m.
\end{aligned}$$

Clearly, under the above assumptions, a λ-d-cover is simply a cover. It is not difficult to see that $\mathcal{A}_1 = \{S_1, S_2, ..., S_{2m}\}$ is an optimal solution, which has cost

$$Opt(I) = c(\mathcal{W}(\mathcal{A}_1)) = c(\{w_0, w_1, ..., w_{2m}\}) = 1 + 2m\varepsilon.$$

However, the output of **GSCA** is $\mathcal{A}_2 = \{S_{2m+1}, S_{2m+2}, ..., S_{3m}\}$, which has cost

$$GSCA(I) = c(\mathcal{W}(\mathcal{A}_2)) = c(\{w_{2m+1}, w_{2m+2}, ..., w_{3m}\}) = 2m.$$

Now, it is clear that, as $\varepsilon \to 0$,

$$\frac{GSCA(I)}{(d(\overline{\mathcal{S}})/d_{min}) \cdot Opt(I)} = \frac{2m}{2m(1+2m\varepsilon)} \to 1,$$

which completes the proof of Theorem 2. ∎

4 Concluding Remarks

4.1 Possible Improvements

When implementing **GSCA**, there are a few things that we can do to improve its performance.

First, notice that the output $\mathcal{A}$ of **GSCA** is a λ-d-cover, but it may not be a minimal one. In other words, it is possible that for some $S_i \in \mathcal{A}$, $\mathcal{A} - \{S_i\}$ is still a λ-d-cover. Therefore, after receiving the output $\mathcal{A}$, we should examine its minimality by checking if $\mathcal{A} - \{S_i\}$ is still a λ-d-cover, for every $S_i \in \mathcal{A}$. Clearly, the output of this procedure will be a minimal λ-d-cover.

Second, notice that **GSCA**'s complexity order is very low, which means that it can handle relatively large problems. To take advantage of this property when dealing with small problems, we make some modifications before starting **GSCA**. Let $\mathcal{S} = \{S_1, S_2, ..., S_n\}$ and $\mathcal{W} = \{W_1, W_2, ..., W_n\}$, as before. Let

$$\mathcal{S}^2 = \{S_i \cup S_j : \text{ for all } i \text{ and } j\} \quad \text{and} \quad \mathcal{W}^2 = \{W_i \cup W_j : \text{ for all } i \text{ and } j\}.$$

Then we can use $(\mathcal{S}^2, \mathcal{W}^2, d, c, \lambda)$ as the input. We have increased the input size, but we can also expect to get a better solution. In general, for each positive integer t, we can define $\mathcal{S}^t$ and $\mathcal{W}^t$ in a similar way and use $(\mathcal{S}^t, \mathcal{W}^t, d, c, \lambda)$ as the input. Depending on our time constraint, we would like to choose the largest possible t. By doing so, we use more of our resources and in return, we should get better solutions.

The last possible improvement comes from the following observation. If $W_i = W_j$ and $\mathcal{A}$ is any λ-d-cover, then, we may assume that, either both S_i and S_j are in $\mathcal{A}$ or neither is in $\mathcal{A}$, since having one in $\mathcal{A}$ makes adding the other to $\mathcal{A}$ free. Therefore, before starting **GSCA**, we can replace S_i and S_j by their union, if $W_i = W_j$. Such a replacement does not change the optimal cost, but it reduces the input size by one. Experiments show that such replacements

do help us getting better solutions, most of the time. In some cases, however, when using **GSCA**, it is also possible that such a replacement can lead us to a worse solution. In the following, we show both cases with two small examples, where $\lambda = 1$ and $d(x) = 1$ for all $x \in S$.

1. In the first case, let $S_1 = \{s_1\}$, $S_2 = \{s_2\}$, $S_3 = \{s_1, s_2\}$, $W_1 = W_2 = \{w_1\}$, $W_3 = \{w_2\}$, $c(w_1) = 2$, and $c(w_2) = 3$. Then $\{S_1, S_2\}$ is an optimal solution, while $\{S_3\}$ is the output of **GSCA**. However, if we merge S_1 and S_2 before starting **GSCA**, then $\{S_1, S_2\}$ will be the output of **GSCA**, which shows that replacing S_1 and S_2 with their union does help us getting a better solution, in this case.

2. In the second case, let $S_1 = \{s_1, s_2, s_3\}$, $S_2 = \{s_4\}$, $S_3 = \{s_1, s_2\}$, $S_4 = \{s_2, s_3, s_4\}$, $W_1 = \{w_1, w_2\}$, $W_2 = \{w_2\}$, $W_3 = W_4 = \{w_1, w_3\}$, $c(w_1) = 1$, $c(w_2) = 2$, and $c(w_3) = 3$. Then $\{S_1, S_2\}$ is the output of **GSCA**, which has cost 3. However, if we merge S_3 and S_4 before starting **GSCA**, then $\{S_3, S_4\}$ will be the output of **GSCA**, which has cost 4. Therefore, in this case, replacing S_3 and S_4 with their union leads us to a worse solution.

Nevertheless, we point out that even though the above modifications may at times produce better solutions, Theorem 2 maintains that they do not improve the bound given in Theorem 1.

4.2 Conclusion

We have defined a generalized set covering problem (GSCP) that extends the classical weighted set covering problem. To approximate an optimal solution to GSCP, we have proposed a highly efficient greedy algorithm (GSCA). In Theorem 1, we have established an upper bound on the ratio of the greedy solution over the optimal solution. We have also proved, in Theorem 2, that our bound is the best possible.

References

[1] S. Arora and M. Sudan. 1997. Improved low-degree testing and its applications. *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, El Paso, Texas, 485-495.

[2] V. Chvatal. 1979. A greedy heuristic for the set covering problem. *Mathematics of operations research*, 4(3):233–235.

[3] U. Feige. 1996. A threshold of $\ln n$ for approximating set cover. *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing*, Philadelphia, Pennsylvania, 314–318.

[4] T. Fujito. 2000., Approximation algorithms for submodular set cover with applications, *IEICE Trans. Inf. & Syst.*, E83-D (3), 480-487.

[5] R. Gandhi, S. Khuller, and A. Srinivasan. 2004. Approximation algorithms for partial covering problems, *Journal of Algorithms*, **4** (1), 55-84.

[6] M. Halldorsson and K. Tanaka. 1996. Approximation and special cases of common subtrees and editing distance. *Proc. 7th Annual International Symposium on Algorithms and Computation*, Lecture Notes in Computer Science 1178, Springer-Verlag, Berlin, 75-84.

[7] D. S. Hochbaum and A. Pathria. Analysis of the greedy approach in problems of maximum k-coverage. *Preprint.*

[8] D. S. Johnson. 1974. Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, 9:256–278.

[9] R. M. Karp. 1975. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, 85–103. Plenum Press, New York.

[10] M. J. Kearns, 1990. *The computational complexity of machine learning*, MIT Press, Cambridge, MA.

[11] L. Lovasz. 1975. On the ratio of optimal integral and fractional covers. *Discrete Mathematics*, 13: 383–390.

[12] R. Raz and S. Safra. 1997. A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, El Paso, Texas, 475-484.

[13] J. D. Ullman, A. V. Aho, and J. E. Hopcroft. 1974. *The Design and Analysis of Computer Algorithms.* Addison-Wesley, Reading, MA.

[14] L. A. Wolsey. 1982. An analysis of the greedy algorithm for the submodular set covering problem, *Combinatorica*, **2**(4) 385-393.