

The Set-Covering Problem

Advanced Algorithms (CSE 794)

Instructor: Tamal K. Dey

Scribe: Rohit Prabhavalkar *

May 29, 2009

1 Introduction

We describe here the set-covering problem and present an approximation algorithm for the problem based on a greedy heuristic. We shall also prove that this algorithm has a logarithmic approximation ratio.

2 The Set-Covering Problem

An instance (X, F) of the *set-covering problem* consists of a finite set X and a set F of subsets of X . We assume that each element of X appears in at least one of the subsets in F ,

$$X = \bigcup_{S \in F} S \quad (1)$$

A set S is said to *cover* each of its elements. The set-covering problem requires us to determine a minimum sized subset of F that covers all the elements of X . In other words, given an instance (X, F) , we are required to find a set $C \subseteq F$ such that,

$$X = \bigcup_{S \in C} S \quad \text{and} \quad |C| \text{ is minimal} \quad (2)$$

An instance of the set-covering problem is presented in Figure (1). As can be seen in the figure, the minimum sized set cover for this instance is 3 (for example, S_2 , S_5 and S_6). The set-covering problem has many practical and useful applications. As an example, consider the problem of choosing a committee of persons such that the members of the committee, as a group, possess all the skills required to complete a particular task. Cost or other considerations may make it desirable to choose a committee of minimum size. This problem may be modeled as follows. Assume that the task requires a set of skills that we denote by X . Each person p who may potentially be selected to appear on the committee possesses some set $S_p \subseteq X$ of these skills. If we define $F = \bigcup_p S_p$, then it is clear that the minimal set cover for this instance (X, F) of the set-covering problem corresponds exactly to the committee of the smallest size that can accomplish the task.

*Department of Computer and Information Sciences, The Ohio State University, Columbus, Ohio, USA.

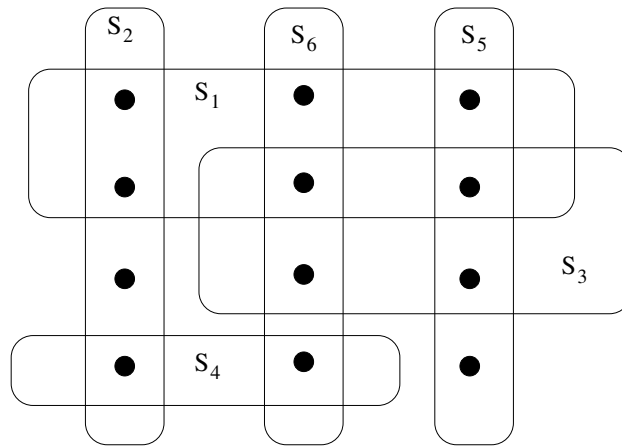


Figure 1: An instance of the set-covering problem. Each dark circle represents an element of X . The set $F = \{S_1, S_2, S_3, S_4, S_5, S_6\}$, with each of the boxes indicating a member of F . The minimal set cover is given by, $C = \{S_2, S_5, S_6\}$.

2.1 NP-Completeness of Set-Covering Problem

We note here that the vertex-cover problem is a special case of the set-covering problem. To see this, assume that we have been given a graph $G = (V, E)$. The vertex-cover problem requires us to determine a minimal set of vertices $V' \subseteq V$ such that every edge $e \in E$ touches atleast one vertex in V' . If we define $X = E$ and $F = \bigcup_{v \in V} E_v$, where E_v is the set of all edges incident on a vertex $v \in V$. It is clear that a minimal set-covering for this instance corresponds exactly to the minimal vertex-cover. Since the decision version of the vertex-cover problem is known to be NP-complete, we conclude that the decision version of the set-covering problem is also NP-complete.

3 A Greedy Approximation Algorithm for the Set-Covering Problem

```

GREEDY-SET-COVER( $X, F$ )
1.  $U \leftarrow X$ ;
2.  $C \leftarrow \phi$ ;
3. while  $U \neq \phi$ 
4.   select  $S \in F$  that maximizes  $|S \cap U|$ ;
5.    $U \leftarrow U - S$ ;
6.    $C \leftarrow C \cup \{S\}$ ;
7. endwhile
8. return  $C$ ;

```

A greedy algorithm for the set-covering problem is presented above. The algorithm maintains a set of elements of X that are as yet uncovered in U . In each iteration of the while loop, the algorithm greedily

chooses the set S that maximally covers the elements of U , until all of the elements of X are covered. The set C contains all the sets that have been chosen as part of the set cover at any point during the operation of the algorithm. Note that by equation (1), it is clear that this procedure terminates with a valid set cover.

3.1 Analysis of GREEDY-SET-COVER

Since each iteration of the while loop in lines 3-7 adds a set $S \in F$ to the cover C , and S must cover at least one of the as yet uncovered elements of X , it is clear that the maximum number of iteration of the while loop is at most $\min(|X|, |F|)$. The actual body of the loop in lines 4-6 can be implemented in time $O(|X||F|)$ and so the overall complexity of GREEDY-SET-COVER is $O(|X||F| \min(|X|, |F|))$ ¹

3.2 Notation

In the following sections we establish the fact that the algorithm GREEDY-SET-COVER, introduced in Section (3) is an approximation algorithm with a logarithmic approximation ratio. We use the following notation, $H(d) = \sum_{i=1}^d 1/i$ and we define $H(0) = 0$.

Theorem 1. *GREEDY-SET-COVER is a polynomial-time $\rho(n)$ -approximation algorithm, where $\rho(n) = H(\max\{|S| : S \in F\})$*

Proof. We have already established that GREEDY-SET-COVER is a polynomial-time algorithm in section (3.1). Let C^* be the minimal set cover and C be the cover determined by the algorithm. We shall assign a cost of 1 unit to each of the sets selected by the algorithm. Let S_i be the i -th set chosen by the algorithm in line 4. We divide the unit cost associated with the selection of this set equally amongst all the elements of X that were covered for the first time by the addition of the set S_i to the set C . We denote this cost by c_x for each element $x \in X$. Note that any element of X is assigned a cost *only once* during the algorithm, when it is covered by some set S for the first time. When a set S_i is added to the collection C by the algorithm, the elements that have already been covered by the previously selected sets are given by $S_1 \cup S_2 \cup \dots \cup S_{i-1}$. Therefore, the number of elements that are covered for the first time is exactly $|S_i - (S_1 \cup S_2 \cup \dots \cup S_{i-1})|$. Since this cost is shared equally between all of these newly covered elements, if x is covered for the first time by S_i then,

$$c_x = \frac{1}{|S_i - (S_1 \cup S_2 \cup \dots \cup S_{i-1})|} \quad (3)$$

Since a cost of 1 was assigned for each set S that was added to the collection C in the algorithm and the algorithm terminates when all elements of X have been covered, we must have,

$$|C| = \sum_{x \in X} c_x \quad (4)$$

To the optimal cover, we assign the cost,

$$\sum_{S \in C^*} \sum_{x \in S} c_x \quad (5)$$

Also, since C^* is a cover, each $x \in X$ appears in at least one of the sets $S \in C^*$. Therefore, we may write

$$\sum_{x \in X} c_x \leq \sum_{S \in C^*} \sum_{x \in S} c_x \quad (6)$$

¹Actually this bound can be improved further. The algorithm can be implemented in time that is linear in the size of the input. For our purposes, however, it is sufficient to note that the algorithm runs in polynomial time.

From equations (4) and (6) we have,

$$|C| \leq \sum_{S \in C^*} \sum_{x \in S} c_x \quad (7)$$

We can now use the result stated below in Proposition (1) to bound the size of the cover returned by the algorithm. Using Proposition (1) we can re-write 7 as,

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

which completes the proof of Theorem 1. □

All that remains, therefore, is to prove the following proposition,

Proposition 1. *If $S \in F$, then $\sum_{x \in S} c_x \leq H(|S|)$*

Proof. Let $S \in F$ be an arbitrary set. We continue to use the same notation as was used previously in the proof of Theorem 1. For each $i = 1, 2, \dots, |C|$ let u_i be the number of elements of S that remain uncovered after S_1, S_2, \dots, S_i have been selected by the algorithm. Therefore,

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

We specially define $u_0 = |S|$. Let k be the smallest index such that each of the elements of S are covered by atleast one of the sets S_1, S_2, \dots, S_k . Note that since the algorithm terminates with each element of X being covered, and that $S \subseteq X$, this index is well defined. Since u_i represents the number of elements of S that are as yet uncovered by S_1, S_2, \dots, S_i , we must have $u_{i-1} \geq u_i$. Also, by construction, $(u_{i-1} - u_i)$ elements of S are covered for the first time by the set S_i for $i = 1, 2, \dots, k$. Therefore, using Equation (3) we have,

$$\sum_{x \in S} c_x = \sum_{i=1}^k (u_{i-1} - u_i) \frac{1}{|S_i - (S_1 \cup S_2 \cup \dots \cup S_{i-1})|} \quad (8)$$

But S_i was chosen in line 4 of the algorithm by the greedy heuristic, since it covered the maximal number of the remaining uncovered elements of X . Therefore we must have,

$$|S_i - (S_1 \cup S_2 \cup \dots \cup S_{i-1})| \geq |S - (S_1 \cup S_2 \cup \dots \cup S_{i-1})| \quad (9)$$

$$= u_{i-1} \quad (10)$$

Note that if Equation (9) did not hold, then the algorithm should have chosen set S instead of S_i at the i -th iteration, since S covers more uncovered elements than S_i . The remainder of the proof consists of a series of algebraic manipulations. From Equations (8) and (10) we have,

$$\begin{aligned}
\sum_{x \in S} c_x &\leq \sum_{i=1}^k (u_{i-1} - u_i) \frac{1}{u_{i-1}} \\
&= \sum_{i=1}^k \sum_{j=u_i+1}^{u_{i-1}} \frac{1}{u_{i-1}} \\
&\leq \sum_{i=1}^k \sum_{j=u_i+1}^{u_{i-1}} \frac{1}{j} && \text{(Since } j \leq u_{i-1}\text{)} \\
&= \sum_{i=1}^k \left(\sum_{j=1}^{u_{i-1}} \frac{1}{j} - \sum_{j=1}^{u_i} \frac{1}{j} \right) \\
&= \sum_{i=1}^k (H(u_{i-1}) - H(u_i)) \\
&= H(u_0) - H(u_k) && \text{(Telescopic sum)} \\
&= H(|S|) - H(0) && \text{(choice of } k \text{ ensures } u_k = 0\text{)} \\
&= H(|S|)
\end{aligned}$$

Hence, proved. □

Finally, the following corollary establishes the fact that the algorithm GREEDY-SET-COVER has a logarithmic approximation ratio.

Corollary 2. *GREEDY-SET-COVER is a polynomial-time $(\ln |X| + 1)$ -approximation algorithm*

Proof. We know that,

$$H(n) = \sum_{i=1}^n \frac{1}{i} \leq \ln n + 1 \quad (11)$$

Therefore, from Theorem 1 and Equation (11), GREEDY-SET-COVER is a polynomial-time algorithm with approximation ratio $\rho(n)$ such that,

$$\begin{aligned}
\rho(n) &= H(\max\{|S| : S \in F\}) && (12) \\
&\leq H(|X|) && (S \subseteq X \Rightarrow |S| \leq |X|) \\
&\leq \ln |X| + 1 && \text{(From Equation (11))}
\end{aligned}$$

This completes the proof. □

4 Concluding Remarks

The algorithm GREEDY-SET-COVER described in Section (3) is an approximation algorithm to the set-covering problem, with a logarithmic approximation ratio.

References

- [1] T. Cormen, C. Leiserson, R. Rivest and C. Stein. Introduction to Algorithms (Second Edition). MIT Press (2001).
- [2] T. Dey. Lecture Notes: CSE 794A: Advanced Algorithms. Ohio State University. Spring 2009.