University of Texas Rio Grande Valley
ScholarWorks @ UTRGV

Theses and Dissertations

5-2017

Randomized Approach tor Set Cover with Multiple Phases*

Ujjwol Subedi
The University of Texas Rio Grande Valley

Follow this and additional works at: https://scholarworks.utrgv.edu/etd

 Part of the Computer Sciences Commons

Recommended Citation

Subedi, Ujjwol, "Randomized Approach tor Set Cover with Multiple Phases*" (2017). *Theses and Dissertations*. 368.
https://scholarworks.utrgv.edu/etd/368

This Thesis is brought to you for free and open access by ScholarWorks @ UTRGV. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of ScholarWorks @ UTRGV. For more information, please contact justin.white@utrgv.edu, william.flores01@utrgv.edu.

RANDOMIZED APPROACH FOR SET COVER WITH MULTIPLE PHASES*

A Thesis

by

UJJWOL SUBEDI

Submitted to the Graduate College of
The University of Texas Rio Grande Valley
In partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

May 2017

Major Subject: Computer Science

RANDOMIZED APPROACH FOR SET COVER WITH MULTIPLE PHASES*

A Thesis
by
UJJWOL SUBEDI

COMMITTEE MEMBERS

Dr. Bin Fu
Chair of Committee

Dr. Zhixiang Chen
Committee Member

Dr. Robert Schweller
Committee Member

May 2017

Copyright 2017 Ujjwol Subedi

All Rights Reserved

ABSTRACT

Subedi, Ujjwol, Randomized Approach for Set Cover with Multiple Phases*. Master of Science (MS), May, 2017, 39 pp., 3 figures, 26 references, 18 titles.

We develop an interactive algorithm for the set cover problem. The algorithm uses multiple stages and select some sets each stage via random samples among the uncovered points. We show that it has a $\mathcal{O}(\log n)$- approximation ratio and takes $\mathcal{O}(\log n)$ rounds and $\mathcal{O}(m^{2+\varepsilon})$ samples each round, where n is the size of universal set and m is the number of sets.

We also prove a $\Omega\left(\sqrt{\frac{logn}{loglogn}}\right)$ lower bound for both the number of phases if each phase has $poly(m)$ the number of samples.

DEDICATION

I would not have completed my Master's degree without the support of my family. I would like to dedicate my work to my family and my daughter Ohana Subedi.

ACKNOWLEDGMENTS

My sincere thanks and regards to my advisor Dr. Bin Fu for his constant support and guidance.

I am also grateful to all the committee members Dr. Zhixiang Chen, Dr. Robert Schweller for carefully reviewing and evaluating my thesis.

TABLE OF CONTENTS

Page

LIST OF FIGURES

Page

CHAPTER I

INTRODUCTION

The set cover problem is a classical NP-hard problem, and one of Karp's twenty-one NP-complete problems [12]. This problem has been extensively studied in the theoretical computer science. The classical set cover problem is the given a set U of elements (called the universe) and a collection S of sets whose union equals the universe U, identify the elements sub-collection of S whose union equals the universe. It can be shown that this algorithm achieves an approximation ration of $ln(n)$ approximation via a greedy approach [11]. The weighted version for set cover with same approximation ratio was extended [2], and a linear programming relaxation achieves the same ratio [14]. Inapproximability results shows that the greedy algorithm is essentially the best possible polynomial time approximation algorithm for set cover [6].

In recent years there is much development of new streaming algorithms for the Set Cover problem, in both theory and applied communities [16] [3] [13] [5] [4] [1] . A streaming model receives elements of each set one by one, and tries to save a small amount of information in a storage. In the streaming Set Cover problem [16], the set of elements U is stored in the memory in advance; the sets $S_1, S_2, ..., S_m$ are stored consecutively in a read-only repository and an algorithm can access the sets only by performing sequential scans of the repository.

We develop an interactive algorithm for the Set Cover problem. The algorithm uses multiple stages and selects some sets each stage via random samples among the uncovered points by those chosen sets. In this model, each set including the universal set is a black box with unknown size. We can only get random samples from those uncovered elements and check the membership if one element belongs to a specific set. It is similar to the classical interactive proof system. Our model is based on communications between a solver and a verifier. The solver provides a list of

sets and a parameter r to the verifier that checks if all elements are covered by the given sets, and will provide r random uncovered elements if some elements are not covered. The complexity for such an interactive model is measured by the number of rounds, and the number of random samples each round. This is motivated by some applications that may be impractical to provide the list of elements in each set, but it is possible to collect random samples from those uncovered elements as they are demanding coverage service.

This is a rigorous theoretical model that needs investigation and has been found some interesting properties. The solver gives an approximation for the problem when it only has partial information about the problem input. The number of random samples tells us how the information a solver need to know. The number of phases measures the depth of computation.

We show that it has $\mathcal{O}(\log n)$- approximation ratio and takes $\mathcal{O}(\log n)$ rounds and $\mathcal{O}(m^{2+\varepsilon})$ samples each round, where n is the size of universal set and m is the number of sets.We also prove a $\Omega\left(\sqrt{\frac{logn}{loglogn}}\right)$ lower bound for both the number of phases if each phase has $poly(m)$ the number of samples.

Our model is based on the applications that the covered elements become silent, and uncovered elements are generating noises for demanding service to be covered. Our model is greatly different from the existing streaming model that needs to scan all elements of all sets. The total time complexity at streaming model is $\Omega(n)$, but the total time complexity in our model is $\mathcal{O}(poly(m)\log n)$.

CHAPTER II

AN OVERVIEW OF GREEDY APPROACH FOR THE SET COVER ALGORITHM

Set Cover Problem is a classical combinatorial optimization problem. Given a set of n elements in the universe X and a family of m sets $S = \{S_1, S_2, ..., S_m\}$, the goal is to find a subset $I \subseteq S$ such that I covers X, i.e. $X \subseteq \cup_{S \in I} S$. Here the number of sets in I has to be as small as possible.

Definition 1. *Let $X = \{x_1, x_2, ..., x_n\}$ be a set of n elements. Let $S = S_1, S_2, ...S_m$ be the collection of m subsets of X such that union of all the subsets of S is X. Let I be the minimal subsets which covers all the elements in X such that $\cup_{j \in I} S_j = X$. A minimum set cover is a set cover I of minimum size, i.e finding minimum subsets to cover all the elements of X is a set cover problem.*

Since Set Cover problem is a classical problem, finding an optimal solution is very hard and is an NP-complete problem. A greedy algorithm is widely used to find best minimal subsets out of all the sets of n elements. The algorithm works perfect and finds the solution very close to an optimal solution but the algorithm does not scale very well to the huge amount of data [3]. Thus, to find a scalable solution for a very large data sets, a tremendous amount of research has been devoted to it.

2.1 Greedy algorithm

Greedy approach is the best-known algorithm for the set cover problem [7]. Greedy approach proceeds greedily, adding one set at a time to the set cover until every element in X is covered. In the first step, it chooses the set that covers the most elements. Once it has been chosen it ignores the set that is already covered and goes to the next step. In the next step, it chooses the set which covers most elements and so on. It continues the step until all the elements have been covered.

1: $I \longleftarrow \phi$
2: Repeat until every element is covered.
3: While $X \neq \phi$
4: do select an $I \in S$ that maximizes $|S \cap X|$
5: $I \longleftarrow I \cup S_j$
6: $X \longleftarrow X \backslash S_j$ // RemovesS_j from X
7: return I

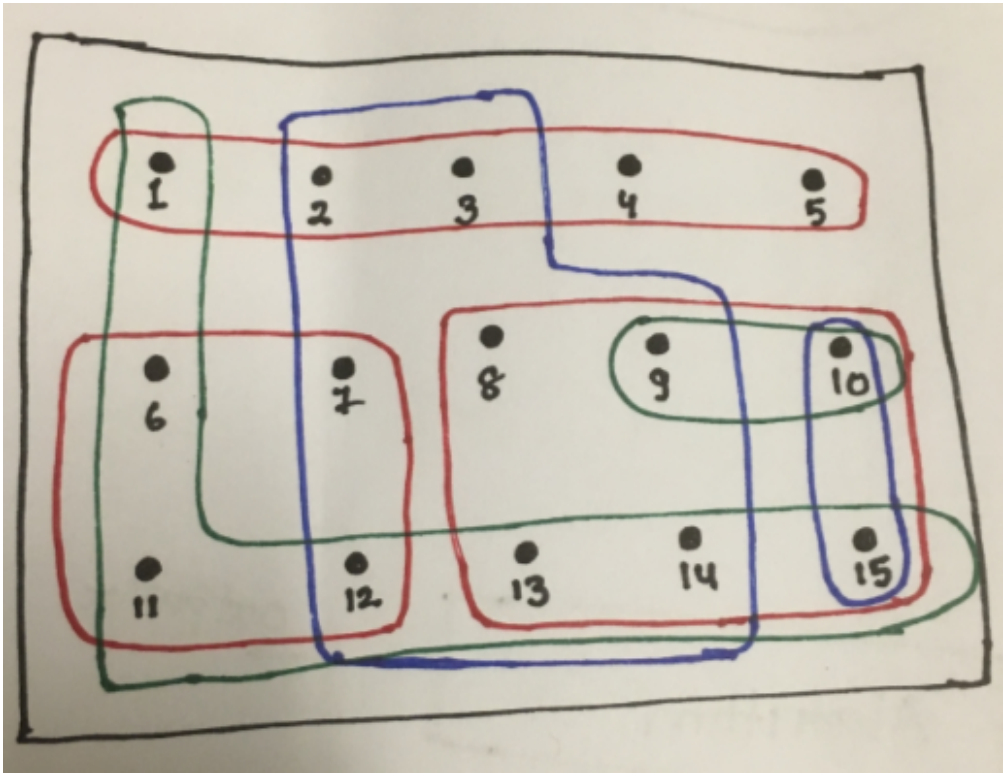

Figure 2.1: A Greedy Approach to Solve the Set Cover Problem

The figure 2.1 consists of 15 elements with 7 subsets.

$U = \{1,2,3,.....,15\}$ and $S = \{S_1, S_2, S_3, S_4, S_5, S_6, S_7\}$.

1st Iteration:

Let UC be the list of uncovered elements.

$C = \phi$

$UC = \{1,2,3,.....,15\}$

Here,

$S_1 = \{1,2,3,4,5\}$

$S_2 = \{2,3,7,8,9,12,13,14\}$

$S_3 = \{10,15\}$

$S_4 = \{9,10\}$

$S_5 = \{8,9,10,13,14,15\}$

$S_6 = \{1,6,11,12,13,14,15\}$

$S_7 = \{6,7,11,12\}$

The optimal solution has size 3 because the union of sets S_1, S_5, S_7 contains all the elements of the universe U.

Our greedy algorithm selects the largest set, $S_2 = \{2,3,7,8,9,12,13,14\}$. Excluding the elements that are already selected will leave us the sets:

$S_1 = \{1,4,5\}$,

$S_3 = \{10,15\}$,

$S_4 = \{10\}$,

$S_5 = \{10,15\}$,

$S_6 = \{1,6,11,15\}$ and

$S_7 = \{6,11\}$.

The greedy now picks the set with maximum number of elements which is

$S_6 = \{1,6,11,15\}$.

Now we are left with sets $\{4,5\}$ and $\{10\}$. In the next step, it picks the set $\{4,5\}$ followed by set $\{10\}$.

Hence, the greedy algorithm produces a set cover $\{S_1, S_2, S_3, S_6\}$ of size 4.

Johnson [11] originally presented the concept of Johnson standard greedy approximation of Set Cover. His algorithm works by picking a subset that covers the most number of remaining uncovered elements at each stage of the algorithm.

Johnson Standard Greedy Algorithm:

Here, let C denotes the covered elements, and UC denotes the uncovered elements and U as

a Universal set which contains n elements and m subsets.

Standard Greedy Set Cover algorithm [11].

Algorithm 1 Johnson Standard greedy SetCover

Set $C = \phi$, UC = U. $I(i) = S_i$, $1 \leq i \leq N$
$UC = \phi$, stop and return.
do select an $j \leq N$ such that $|I(j)|$ is maximized.
Set $C \longleftarrow C \cup S_j$, $UC = UC - I(j)$,
$I(i) = I(i) - I(j)$, $1 \leq i \leq N$
Go to 2.

Example 1:

Let, $U = \{1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16\}$

$S_1 = \{1,2,3,4,5,6,7,8\}$

$S_2 = \{6,7,8,10,11,12,13\}$

$S_3 = \{1,4,7,10,12,14\}$

$S_4 = \{2,5,7,8,11,13,14\}$

$S_5 = \{3,6,9,15,16\}$

$S_6 = \{13,14,15\}$

First Iteration:

$C = \phi$, which means we do not have any elements covered yet.

$UC = \{1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16\}$

Since $UC \neq \phi$,

For $j = 1$ I(1) = 8 has maximum cardinality

$C = \{S_1\}$

So, $UC = UC - C$

$UC = \{9,10,11,12,13,14,15,16\}$

$S_2 = \{10,11,12,13\}$

$S_3 = \{10,12,14\}$

$S_4 = \{11,13,14\}$

$S_5 = \{9,15,16\}$

$S_6 = \{13, 14, 15\}$

Second Iteration: $UC \neq \phi$, so we continue,

For $j = 2$ I(2) = 4 has maximum cardinality

$C = \{S_1, S_2\}$

So, $UC = UC - C$

$UC = \{9, 14, 15, 16\}$

$S_3 = \{14\}$

$S_4 = \{14\}$

$S_5 = \{15, 16\}$

$S_6 = \{14, 15\}$

Third Iteration: $UC = \phi$, so we continue,

For $j = 5$ I(5) = 2 has maximum cardinality

$C = \{S_1, S_2, S_5\}$

So, $UC = UC - C$

$UC = \{14\}$

$S_3 = \{14\}$

$S_4 = \{14\}$

$S_6 = \{14\}$

Fourth Iteration: Since, $UC = \phi$, so we continue,

For $j = 3$ I(3) = 1 has maximum cardinality,

$C = \{S_1, S_2, S_3, S_5\}$

$UC = \phi$.

Fifth Iteration:

Since $UC = \phi$, we stop here ,

$C = \{S_1, S_2, S_3, S_5\}$

Hence, Standard greedy algorithm produces a set cover $\{S_1, S_2, S_3, S_5\}$ of size 4. But the optimal solution for the set cover is $\{S_3, S_4, S_5\}$ of size 3.

2.2 Approximation Algorithm

Approximation algorithm is a technique which helps to find the solution within a factor of optimum solution. In our algorithm we have shown that our approximation ratio is $\mathcal{O}(\log n)$.

2.3 Randomization Algorithm

A randomized algorithm is a technique that employs a degree of randomness as a part of its logic. It has been used very often and is one of the powerful randomized technique in standard algorithm to reduce either time complexity or space complexity. Random numbers r are generated while using randomized algorithm within a specified range of numbers, and hence making a decision based on r value. Here in this paper, chernoff inequality and hoeffiding inequality plays an important role in designing and analyzing the accuracy of our algorithm. It shows how the number of samples determines the accuracy of the approximation.

2.4 Maximal K-coverage for Set Cover

Let U be the universal set of n elements and $S_1, S_2, ..., S_m$ be the subsets of S set such that $1 \leq i \leq m$, and $S_i \subseteq U$. Here in maximum k-coverage problem, we are given an integer k and the goal is to find at most k sets from S such that the size of their union is maximum.

Definition 2. *Given a set U of elements n and a collection of subsets $S_1, S_2, ..., S_m$ such that $1 \leq i \leq m$ and an integer k. The maximum k-coverage problem finds the k sets such that the size of their union is maximum.*

CHAPTER III

WEIGHTED SET COVER

A set cover $C \subseteq$ is a subcollection of the sets whose union is U, given a collection S of the sets over a Universe U. Finding a minimum cardinality set, given S would be the set cover problem. For the *weighted set-cover* problem, a weight $w_s \geq 0$ is specified for each set $s \in S$. The goal then would be to find a set cover C of minimum total weight $\sum_{s \in C} \ w_s$.

By choosing a set s that minimize the weight w_s divided by number of elements in s not covered by chosen sets, the greedy algorithm builds a cover for weight set cover.

Theorem 1. *[18] The greedy algorithm returns a set cover of weight at most H_k times the minimum weight of any cover.*

Proof. In the optimal set cover C^*, suppose the greedy algorithm covers the elements of s, where $s = \{x_k, x_{k-1}, ..., x_1\}$, in the following order $x_k, x_{k-1}, ... $. Initially the algorithm covers elements x_i of s, then, at least i elements of s are not covered. The idea is that the greedy algorithm chooses s in such an iteration so that it would pay w_s/i per element covered. Therefore, in a way the greedy algorithm charges w_s/i per element x_i to be covered. The total amount charged, to elements in s after summing over i, is $w_s H_k$ at the greatest. Noting that every element is in some set in C^*, summing over $s \in C^*$, the total amount charged to elements is at most $\sum_{s \in C^*} \ w_s H_k = H_k OPT$ $\square$

The theorem was shown first for the unweighted case (each $w_s = 1$) by Johnson [11] , Lova'sz [14], and Stein [17], then extended to the weighted case by Chva'tal [2].

CHAPTER IV

COMPUTATIONAL MODEL AND COMPLEXITY

In this section, we show our model of computation, and the definition of complexity. Assume that A_1 and A_2 are two sets. Their union $A_1 \cup A_2$ contains the elements in either A_1 or A_2. Define A_2 - A_1 to be the sets of elements in A_2, but not in A_1. Define their intersection $A_1 \cap A_2$ to be the set of elements in both A_1 and A_2. For example, A_1 = {3, 5} and A_2 = {1, 3, 7}, then $A_1 \cup A_2$ = {1, 3, 5, 7}, A_2 - A_1 = {1, 7}, and $A_1 \cap A_2$={3}. For a finite set A, we use $|A|$, *cardinality* of A, to be the number of distinct elements in A. For a real number x, let $\lceil x \rceil$ be the least integer $y \geq x$, and $\lfloor x \rfloor$ be the largest integer $z \leq x$. For example,$\lceil 3.2 \rceil = 4$, and $\lfloor 3.2 \rfloor = 3$. Let N = {0, 1, 2,.....} be the set of nonnegative integers, $R = (-\infty, +\infty)$ be the set of all real numbers, and $R^+ = [0, +\infty)$ be the set of all nonnegative real numbers. An integer s is a $(1+\varepsilon)$-approximation for $|A|$ if $(1-\varepsilon)\ |A| \leq (1+\varepsilon)\ |A|$.

Definition 3. *The randomized computation for set cover is defined below: An input L is a list of sets A_1, A_2,....., A_m that support the following operations:*

1. *Function RandomUncoveredElement (L', L) returns a random element x that is not covered by the sets in sublists L' of L.*
2. *Function Query(x, A_i) returns 1 if $x \in A_i$, and 0 otherwise.*
3. *Each phase of the algorithm can select some sets and put them into a list L', which is empty in the beginning.*
4. *It returns L' as an (approximate) set cover solution at the end of the algorithm.*

Definition 4. *Let $A(.,.)$ be a randomized algorithm for the set cover problem in the model given by definition 3. Let L be a list of input sets. We use the tuple $(T(.),R(.))$ to characterize the computational complexity, where*

- *$A(L,p)$ is a deterministic computation at path p.*
- *$T(.)$ is a function for the number of phases such that each phase allows to generate at most $R(m,n)$ random samples from those uncovered elements.*
- *$R(.)$ is a function to be the upper bound of the number of the random samples each phase via $RandomUncoveredElement(L')$.*
- *The algorithm $A(L,.)$ gives $f(n)$-approximation if it returns a sublist L' of sets from L in at least 3/4 paths p such that $|L'| \leq f(n).|opt(L)|$, where $opt(L)$ is a sublist of sets in an optimal solution.*

4.1 Interactive Proof System

Definition 5. *[10] For the given problem, interactive proof system works in the following ways:*

- *we have two participants, a verifier V and a solver SV.*
- *Both of the participants are given the same input.*
- *V and SV exchange messages for the given problem.*
- *Both parties performs some computation on the problem and try to solve it using local randomness.*
- *Eventually V verifies whether particular problem is solved. Verifier has to be convinced and satisfied with the solution the solver has solved.*

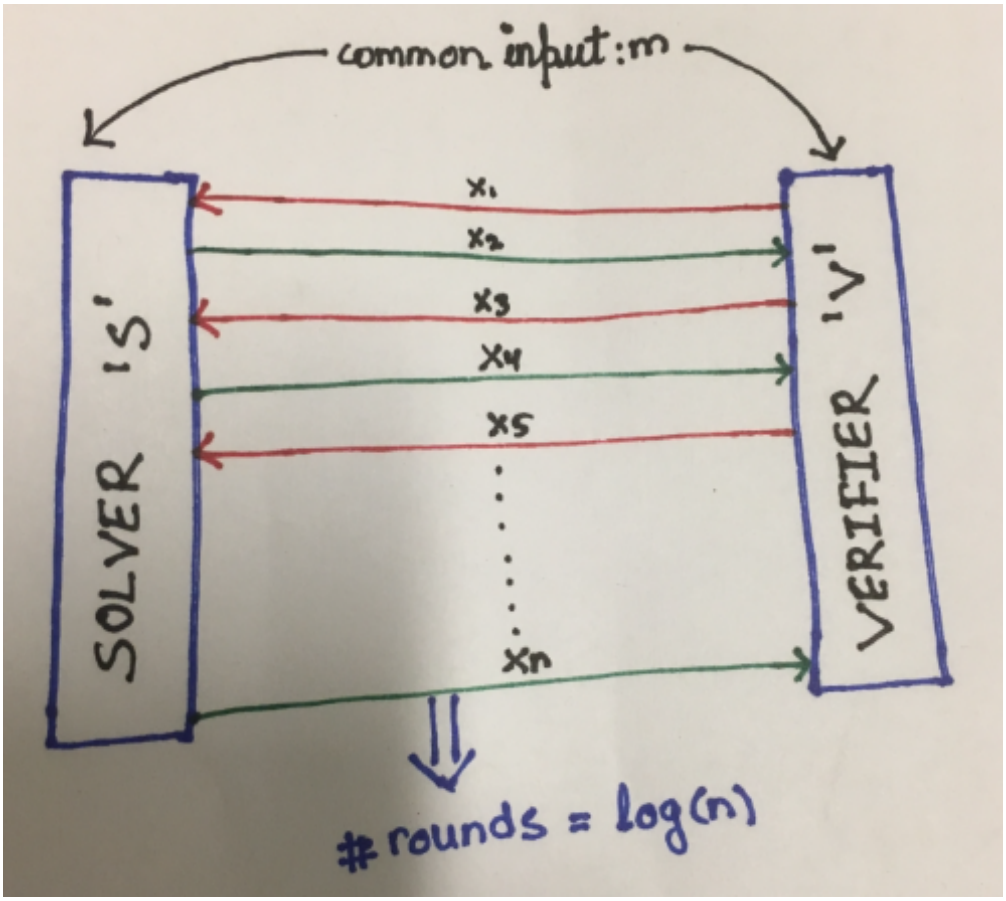

Figure 4.1: Interactive Proof System

The main goal of the solver is to convince and satisfy the verifier of its validity and verifier's purpose is to embrace only a correct solution [10]. Interactive proof system can be interpreted in many ways in a real life. It is just a process of verifying a problem solved by a solver. For instance, a thesis is a process of doing extensive research and coming up with your own solution for a given problem. Let's say, one of the students from University of Texas Rio Grande Valley in computer science department wants to work on a thesis project. To work on his thesis he has to first choose an adviser. Once he chooses the adviser he starts working with the adviser and tries to solve the problem given by the adviser. Here adviser acts as a "verifier" whereas student as a "solver". He asks student to solve some problem and gives some random hint to solve it. Once the student receives the problem with the hint, he tries to solve it and solved problem is sent back to adviser. Adviser verifies it and gives some more problem to solve it in order to fulfill the requirements of his

thesis paper. This process continues until all the problems needed to finish the thesis are solved and verified. Goldwasser, Micali and Rackoffin came up with the concept of interactive proofs as a game between "solver" and a "verifier" in the paper [8].

An interactive proof system is said to be good proof system for a solution if it meets the following properties:

1. **Completeness**: Verifier always accepts the solution if it is true.
2. **Soundness**: Verifier always rejects the solution if it is false.
3. **Efficient**: The strategy used by a verifier to verify the solution is effective and very efficient (runs in polynomial time in the *NP* case).

We developed an interactive algorithm for the set cover problem. In our model, function $RandomUncoveredElement(L', L)$ acts as a verifier "V" and function $Query(x, A_i)$ acts as a solver "SV".

Example:

Let U = {1, 2, 3, 4,....., 20} be the universal set. Let $S = \{S_1, S_2, S_3, S_4, S_5\}$ be the subsets of U where,

$S_1 = \{1, 2, 3, 4, 5, 6\}$

$S_2 = \{4, 5, 6, 7, 8, 20\}$

$S_3 = \{7, 8, 9, 10, 11, 12, 13, 14, 15\}$

$S_4 = \{4, 5, 16, 17, 18\}$

$S_5 = \{1, 2, 3, 19, 20\}$

We have five sets $S = \{S_1, S_2, S_3, S_4, S_5\}$ with $U = \{1, 2, 3, 4, ..., 20\}$. According to our computational model, Input $L = \{S_1, S_2, S_3, S_4, S_5\}$ is a list of sets $\subseteq U$. Let r be the random samples.

Solver "SV" is given with the input list U. It sends the list to the verifier "V" to check if the list U is covered. Here in our model, covered elements remain silent whereas uncovered one remain noisy demanding to be covered.

Once the input list is sent to the V, it verifies whether the elements of the list are covered. If not, V provides the list and the random samples r to the solver.

Let random samples provided to the SV is $\{1,5,6,7,8,9,10,15,18,19\}$. With the help of random samples r =, the solver solves it. Function $Query(x,A_i)$ acts as a Solver "SV".

$$Query(x,A_i) = \begin{cases} 1, & \text{if } x \in A_i \\ 0, & \text{otherwise} \end{cases}$$

Here, from the given random samples r, solver tries to find the set the samples represent. In the process it finds that,

$1 \in \{S_1, S_5\}$

$5 \in \{S_1, S_2\}$

$6 \in \{S_1, S_2\}$

$7 \in \{S_2, S_3\}$

$8 \in \{S_2\}$

$9 \in \{S_3\}$

$10 \in \{S_3\}$

$15 \in \{S_3\}$

$18 \in \{S_4\}$

$19 \in \{S_5\}$

Then, $\{1,5,6,7,8,9,10,15,18,19\}$ is added to the set and is covered. Once it is covered these elements remain silent. SV sends back the list of sets to V to verify if all the elements are covered. V verifies it. Since the sets are not covered yet, it sends back to solver the list of sets and random samples r from uncovered areas (elements).

Let the $r = \{2,3,4,13,14\}$.

SV repeats the same process to solve it. In the process it finds that,

$2 \in \{S_1, S_5\}$

$3 \in \{S_1, S_5\}$

$4 \in \{S_1, S_2, S_4\}$

$13 \in \{S_3\}$

$14 \in \{S_3\}$

Once the process is done, the list of the elements is added to the sublist L'. Hence, $\{2,3,4,13,14\}$ remain silent. *SV* sends back the list of elements to V to verify if all the elements of the sets are covered. V verifies it.

Here, elements of set $S_1 = \{1,2,3,4,5,6\}$ are all selected and the set S_1 is said to be covered and we put the set in sublist L'. So, the elements of S_1 remain silent.

Again, V provides the *SV* with input list of sets and a random samples r from and uncovered elements. Let $r = \{11,12,16\}$. Once the solver *SV* receives the list of sets with some random samples, it repeats the same process as described above to process it. In the process it finds that,

$11 \in \{S_3\}$

$12 \in \{S_3\}$

$16 \in \{S_4\}$

So, $\{11,12,16\}$ are added to the set and these elements once selected remains silent.

This process continues until all the elements of the input list U is covered. In every phase in our computational model, at least half of the elements are covered. Hence, our algorithm takes $\mathcal{O}(\log n)$ rounds.

4.2 Deterministic Computation

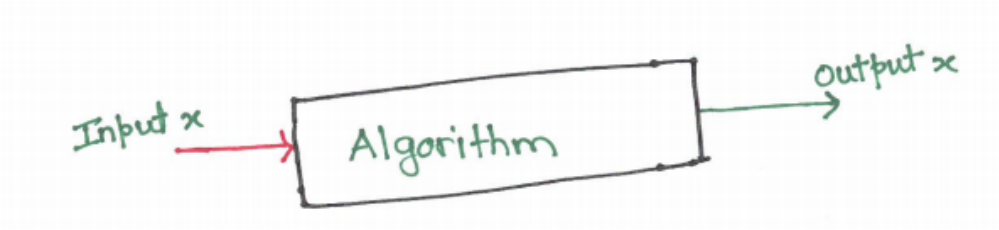

Figure 4.2: Deterministic Algorithm

Deterministic Algorithm is an algorithm which always produces same output, when particular input is provided. While defining in terms of state machine, it is defined as a state machines that behaves in a predetermined sequence of actions from one state to another state of the machine. If the machine is deterministic, just by entering the input the machine will be in its starting state or initial state. The current state determines its upcoming state and will never stop or finish until it delivers the same output (automatically follows a predetermined sequence of operations). Deterministic Finite Automaton and Deterministic Turing Machine are two examples of abstract machine which are deterministic. In our model we have used deterministic algorithm on a path p.

CHAPTER V

OUTLINE OF OUR METHODS

For the two sets A and B, we develop a randomized method to approximate the cardinality of the difference $B-A$. We approximate the size of $B-A$ by sampling a small number of elements from B and calculating the ratio of the elements in $B-A$ by quering the set A. The approximate $|A \cup B|$ is the sum of an approximate $|A|$ and an approximate of $|B-A|$.

A greedy approach will be based on the approximate difference between a new set and the union of sets already selected. Assume that A_1, A_2,....., A_n is the list of sets for the set cover problem. After A_{i_1},....., A_{i_t} have been selected, the greedy approach needs to check the size $|A_j - (A_{i_1} \cup A_{i_2} \cup ..... \cup A_{i_t})|$ before selecting the next set. Our method to estimate $|A_j - (A_{i_1} \cup A_{i_2} \cup ..... \cup A_{i_t})|$ is based on randomization in order to make the time independent of the sizes of input sets. Some random samples are selected from set A_j.

During the accuracy analysis, Hoeffiding Bound (see [9]) plays an important role. It shows how the number of samples determines the accuracy of approximation.

Theorem 2. *Let $X_1, ..., X_s$ be s independent random 0-1 variables and $X = \sum_{i=1}^{s} X_i$.*

- *if X_i takes 1 with probability at most p for i=1,.....,s then for any $\varepsilon > 0$, $Pr(X > ps + \varepsilon s) < e^{-\frac{1}{2}s\varepsilon^2}$.*

- *if X_i takes 1 with probability at least p for i=1,.....,s then for any $\varepsilon > 0$, $Pr(X < ps + \varepsilon s) < e^{-\frac{1}{2}s\varepsilon^2}$*

We define the function $\mu(x)$ in order to simply the probability mentioned in theorem

$$\mu(x) = e^{-\frac{1}{2}s\varepsilon^2} \tag{5.1}$$

During the accuracy analysis, Chernoff Bound (see[15]) plays an important role. It shows how the number of samples determines the accuracy of approximation.

Theorem 3. *Let $X_1, ..., X_s$ be s independent random 0-1 variables, where X_i takes 1 with probability at least p for i=1,.....,n. Let $X = \sum_{i=1}^{n} X_i$, and $\mu = E[X]$. Then for any $\delta > 0$, $Pr(X < (1-\delta)pn) < e^{-\frac{1}{2}\delta^2 pn}$.*

Theorem 4. *Let $X_1, ..., X_s$ be s independent random 0-1 variables, where X_i takes 1 with probability at most p for i=1,.....,n. Let $X = \sum_{i=1}^{n} X_i$, and $\mu = E[X]$. Then for any $\delta < 0$, $Pr(X > (1+\delta)pn) < \left[\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}}\right]^{pn}$.*

Define $g_1(\delta) = e^{-\frac{1}{2}\delta^2}$ and $g_2(\delta) = \left[\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}}\right]$. Define $g(\delta) = max(g_1(\delta), g_2(\delta))$. We note that $g_1(\delta)$ and $g_2(\delta)$ are always strictly less than 1 for all $\delta > 0$. It is trivial for $g_1(\delta)$. For $g_(\delta)$, this can be verified by checking that the function $f(x) = (1+x)ln(1+x) - x$ is increasing and $f(0) = 0$. This is because $f'(x) = ln(1+x)$ which is strictly greater than 0 for all $x > 0$.

A well-known fact in probability theory is the union bound that is represented by the inequality

$$Pr(E_1 \cup E_2 \cup ..... \cup E_m) \leq Pr(E_1) + Pr(E_2) + ... + Pr(E_m), \tag{5.2}$$

where E_1, E_2,....., E_m are m events that may not be independent. In the analysis of our randomized algorithm, there are multiple events such that the failure from any one of them may fail the entire algorithm. We often characterize the failure probability of each of those events, and use the above inequality to show that the whole algorithm has a small chance to fail, after showing that each of them has a small chance to fail.

CHAPTER VI

ALGORITHM FOR SET COVER

In this section, we describe an interactive algorithm for the set cover problem. The number of rounds and the number of samples are analyzed.

Definition 6. *Let $R = x_1, x_2, ..., x_w$ be a list of elements, B be a set, and let L be a list of sets $A_1, A_2, ..., A_u$. Define Test(L,B,R) =* $|(j : 1 \leq j \leq w, x_j \in B$, *and* $x_j \notin (A_1 \cup A_2 \cup ..... \cup A_u)|$

The algorithm ApproximateDifference(.) gives an approximation s for the size of $B - A$. It is very time consuming to approximate $|B - A|$ when $|B - A|$ is much less than $|B|$. The algorithm ApproximateDifference(.) returns an approximate value s for $|B - A|$ with a range $[(1 - \delta)|B - A|, (1 + \delta)|B - A|]$, and will not lose much accuracy when it is applied to approximate the set cover by controlling the two parameters δ and ε.

Algorithm Test(L, B, R)

Input: L is a list of sets $A_1, A_2, ..., A_u$, B is another set, R is a set of elements in U.

1. Let $x_1, ..., x_w$ be the items in R;
2. For $i = 1$ to w
3. Let $y_i = 1$ if $x_i \in B$ and $(x_i \notin A_1 \cup A_2 \cup ..... \cup A_u)$, and 0 otherwise;
4. Return $t = y_1 + ... + y_w$;

End of Algorithm

We show a multi stage algorithm for the set cover problem. It uses the randomized algorithm to estimating the set difference.

Algorithm: SetCover(L, ε, γ)

Input: L is a list of sets $A_1, A_2, ..., A_m$ and ε and γ are real parameters in (0,1).

Steps:

1. Let $L' = \phi$;
2. Let $\alpha = \frac{1}{2}$;
3. Let $i = 1$;
4. Stage i:
5. Get a set R of $g(m) = m^{2+\varepsilon}$ random samples that are not covered by the sets in L'.
6. If R does not contain any element, then stop the algorithm (all elements are covered).
7. Repeat

{

8. For each set A_j in L but not in L', let t_j = Test(L', A_j, R);
9. If there is a set A_j that has the largest t_j and $t_j \geq \frac{\alpha g(m)}{4m}$,
10. then put A_j into L', and remove it from L;
11. else terminate loop, and enter stage $i+1$ (and let $i = i+1$);

}

Return L'

End of the Algorithm

Theorem 5. *There is a* $\mathscr{O}(\log n, m^{2+\varepsilon})$ *interactive algorithm for the set cover with* $\mathscr{O}(\log n)$*-approximation ratio.*

Proof. Our analysis is only at phase i, which may select several sets. Since the random samples are from those uncovered point, the already selected sets and covered points can be ignored.

Let $\gamma = \frac{1}{4m^2 2^m}$. Let $g(m)$ be the number of random points generated in the beginning of each phase. We let $\alpha = \frac{1}{2}$, and $\beta = \frac{\alpha}{9}$.

Let V_i be the set of elements that have been uncovered by the sets selected after stage i. Let U_t be the set of uncovered elements after selecting t sets. Let T_t be the union of the selected sets after selecting t sets. Assume that there are at least $\alpha \cdot |U_t|$ uncovered by T_t. There exists at least one set S_i such that S_i covers at least $\frac{\alpha \cdot |U_t|}{m}$ points. □

Claim 1. *At stage i, for all S_u with $p_u \geq \frac{\beta}{m}$, with probability at most $\gamma \leq \frac{1}{4m}$, we do not have all of the following conditions:*

$$Test(L', S_u, R) \geq (1-\delta) p_u |R|, \tag{6.1}$$

$$Test(L', S_u, R) \leq (1+\delta) p_u |R|, \tag{6.2}$$

$$Test(L', S_u, R) > \frac{\alpha g(m)}{4m}, \tag{6.3}$$

Proof. Define p_i to be the probability that a random item of R is in $S_i - T$. We have there is a set S_i with $p_i \geq \frac{\alpha}{m}$. Assume $p_i \geq \frac{\beta}{m}$. We have the inequalities:

$$\mu(\delta)^{p_i g(m)} \leq \mu(\delta)^{\frac{\beta}{m} \cdot g(m)} \leq \gamma \leq \frac{1}{4m^2 2^m} \tag{6.4}$$

For a S_u with $p_u \geq \frac{\beta}{m}$, with probability at most $\gamma \leq \frac{1}{4m^2 2^m}$, we do not have all of the following conditions:

$$Test(L', S_u, R) \geq (1-\delta) p_u |R|, \tag{6.5}$$

$$Test(L', S_u, R) \leq (1+\delta) p_u |R|, \tag{6.6}$$

$$Test(L', S_u, R) > \frac{\alpha g(m)}{4m}, \tag{6.7}$$

There are at most 2^m cases of subsets selected in the process of algorithm. Thus, we probability at most $Q = m \cdot 2^m \cdot \frac{1}{4m^2 2^m} \leq \frac{1}{4m}$, at least one of S_u with $p_u \geq \frac{\beta}{m}$ fails some inequalities among (6.1) to (6.3). We assume that inequalities (6.1) to (6.3) are true for each S_u with $p_u \geq \frac{\beta}{m}$. By the conditions at line 9 in SetCover(.), one set will be selected.

□

Claim 2. *At stage i, if $|U_{t-1}| \leq \alpha |V_{i-1}|$, then there is a (t-th) set that will be selected with failure probability at most $\frac{1}{4m}$.*

Proof. If $|U_{t-1}| \leq \alpha |V_{i-1}|$, then there is a set S_u with $p_u \geq \frac{\alpha}{m} \geq \frac{\beta}{m}$. By the conditions at line 9 in SetCover(.) and inequality (6.3) in claim 1, with probability at most $\gamma \leq \frac{1}{4m}$, one set will not be selected.

With $g(m)$ samples each round, the set with the largest uncovered elements has $\Omega(m^{1+\varepsilon})$ elements from R since there are m sets available. Let $S_i - T$ be the largest set among all $S_a - T$. □

Claim 3. *If S_j is selected, then $p_j \geq \frac{1}{2} \cdot p_i$ with failure probability at most $\frac{1}{4m}$, where S_i is the set such that $|S_i - T_t|$is the largest.*

Proof. In the case that S_j is selected, then by the line 8 in SetCover(.), we have $Test(L', S_j, R) \geq \frac{\alpha g(m)}{4m}$. If $p_j < \frac{\beta}{m}$, with probability at most $\mu(1)^{\frac{\beta}{m} g(m)} \leq \frac{1}{4m^2 2^m}$, $Test(L', S_j, R) \geq \frac{2\beta g(m)}{m}$. We know $\frac{2\beta g(m)}{m} < \frac{\alpha g(m)}{4m}$.

The Set S_i with the largest probability is at least $\frac{\alpha}{2m} > \frac{\beta}{m}$. We have $Test(L', S_i, R) \geq (1 - \delta) p_i |R|$. As S_j is selected according to the conditions in line 9 of SetCover(.), we have $(1 + \delta) p_j |R| \geq (1 - \delta) p_i |R|$. Therefore, $p_j |R| \geq \frac{1-\delta}{1+\delta} \cdot p_i |R| \geq \frac{1}{2} \cdot p_i |R|$.

Therefore, when each S_j is selected, the number of uncovered points is reduced by at least $\frac{1}{2m_o}$ factor, where m_o is the optimal number of sets to cover all points. The number of remained items is at most $z(1 - \frac{1}{2m_o})$, where z is the number of uncovered items before S_j is chosen.

After k sets have been selected, we have the number of uncovered items bounded by $n(1 - \frac{1}{2m_o})^k$, where n is the total number of points to be covered in the beginning. Therefore, when

$k = c \cdot m_o \cdot logn$ for some fixed $c > 0$. We have $n(1 - \frac{1}{2m_o})^k < 1$, which implies that all points are covered after selecting k sets. Thus, the approximation factor is $\mathscr{O}(\log n)$.

If no set is selected in a stage, it keeps the same result from the last stage. Since each stage has probability at most Q with no set being selected when it is not fully covered, the probability that one stage has no set being selected is most $mQ \leq \frac{1}{4}$ among the first m stages.

If no stage fails, each stage can reduce the number of uncovered elements by at least half. Therefore, it takes $\mathscr{O}(\log n)$ rounds. □

CHAPTER VII

LOWER BOUND FOR SET COVER

In this section, we derive lower bound for the number of rounds for set cover problem. If the number of rounds is not sufficient, it is unlikely to have good approximation ratio.

7.1 Randomized Model for Lower Bounds

We define a randomized computation model for our lower bound.

Definition 7. *A randomized computation* $T(.,.)$ *for the maximum coverage problem is a tree* T *that takes an input of list* L *of finite sets. Let* U *be the universal set, which is the union of all sets in the list* L*.*

1. *Each node of* $T(L,.)$ *(with input list* L *of sets for set cover problem) allows an operation defined in definition 1.*

2. *A branching point of* p *that has* s *children* $p_1, p_2, ..., p_s$ *and is caused by the following two cases*

   - *RandomUncoveredElement() returns a random element in* $U - S = \{a_1, a_2, ..., a_u\}$ *such that* p_j *is the case that* a_j *is selected, Where* S *is the list of selected sets.*
   - *RandomNumber(s) returns a random element in* $\{0, 1, ..., s-1\}$ *for an integer* $s \geq 0$ *such that* p_j *is the case that* $j+1$ *is returned.*

3. *A computation path is determined by a series of numbers* $r_0, r_1, r_2, ..., r_t$ *such that* r_j *corresponds to the* j^{th} *branching point for* $j = 1, 2, ..., t-1$ *and* r_0 *is the root, and* r_t *is a leaf.*

4. *A partial path p is an initial part of a path that starts from the root r_0 of computation to a node q*

5. *The root node r_0 has weight $w(r_0) = 1$.*

6. *If a partial path p from root r_0 to a node q that has children $p_1, ..., p_s$ and weight $w(q)$. Then $w(p_1) = w(p_2) = ... = w(p_s) = \frac{w(q)}{s}$, where $w(p_i)$ is the weight for p_i.*

7. *The weight of a path from the root r_0 to a leaf q has a weight $w(q)$, which is the weight of q.*

8. *The output of the randomized computation$T(L,.)$ (with input L) on a path p is defined to be $T(L,p)$.*

The weight function $w(.)$ determines the probability of a partial path or path generated in the randomized computation. In Definition 7, we give the concept of a shared path for randomized computation under two different inputs of lists of sets. Intuitively, the computation of the two shared paths with different inputs does not have any difference, gives both the same output, and has the same weight.

Definition 8.

- *Let L be a list of sets $A_1, ..., A_m$, and L' be another input list of m sets $A'_1, ..., A'_m$. If $|A_i| = |A'_i|$ for $i = 1, 2, ..., m$, then L and L' are called equal size of list of sets.*

- *Let L be a list of sets $A_1, ..., A_m$, and L' be another input list of m sets $A'_1, ..., A'_m$ such that they are of equal size. A partial path p is shared by $T(L,.)$ and $T(L',.)$ if*

  - *path p gets the same result for each random access to RandomUncoveredElement () and*
  - *path p gets the same result for each random access to RandomNumbers(s).*

Let P be a set of path in T(L,.), define $W(P)$ be the sum of weights of all paths in P. In other words, $W(P) = \sum_{p \in P} w(p)$. The algorithm $T(L,.)$ gives an $f(n)$-approximation if there is a set P of paths in $T(L,.)$ of weight $W(P)$ at least $\frac{3}{4}$ such that for each $p \in P$, $|L'| \leq f(n) \cdot |opt(L)|$, where $L' = T(L,p)$ and $opt(L)$ is a sublist of sets in an optimal solution.

The weight function $w(.)$ determines the probability of partial path or path generated in randomized computation.

7.2 Lower Results

The idea to derive a lower bound for the number of phases is to construct two lists L_1 and L_2 that have the same solution in most cases by running the algorithm, but greatly different number of sets for the set cover solutions.

Lemma 1. *Let $g(m,n)$ and $f(m,n)$ be a functions from $N \times N$ to N. Let $g(.)$ be defined by the following recursions:*

$$g(1,m,n) = 2, \tag{7.1}$$

$$g(i,m,n) = 10f(m,n)(1 + \sum_{j=1}^{i-1} g(j,m,n)), \tag{7.2}$$

Then $g(i,m,n) \leq (10(f(m,n)+3))^i$.

Proof. By equation(7.2), we have $1 + \sum_{j=1}^{i-2} g(j,m,n) \leq \frac{1}{f(m,n)} g(i-1,m,n))$. Therefore, we have

$$g(i,m,n) \leq f(m,n)(g(i-1,m,n) + \frac{1}{f(m,n)} g(i-1,m,n)) + 1 \tag{7.3}$$

$$\leq 10(f(m,n)+1)(g(i-1,m,n) + \frac{1}{f(m,n)} g(i-1,m,n)) \tag{7.4}$$

$$\leq 10(f(m,n)+1)(1 + \frac{1}{f(m,n)} g(i-1,m,n)) \tag{7.5}$$

$$\leq 10(f(m,n)+3)(g(i-1,m,n)) \tag{7.6}$$

$$\leq (10(f(m,n)+3))^i (g(1,m,n)) \tag{7.7}$$

$$\leq (10(f(m,n)+3))^i \tag{7.8}$$

□

Lemma 2. *Let $c(m,n), g(m,n), h(m,n) and f(m,n)$ be the functions from $N \times N$ to N. Let $h(.)$ be defined by the following recursions:*

$$h(k,m,n) = 1, \tag{7.9}$$

$$h(i,m,n) = 100c(m,n)\cdot s(m,n)\sum_{j=i+1}^{k} g(j,m,n)h(j,m,n) \tag{7.10}$$

Then $h(i,m,n) \leq (100ec(m,n)s(m,n))^k(10(f(m,n)+3))^{k^2}$, *where* $g(.)$ *is given in Lemma 1.*

Proof. By equation (7.10), we have $\sum_{j=i+2}^{k} g(j,m,n)h(j,m,n) \leq \frac{1}{100c(m,n)s(m,n)}h(i+1,m,n)$. Therefore we have,

$$h(i,m,n) \leq 100c(m,n)s(m,n)(g(i+1,m,n)h(i+1,m,n)) \tag{7.11}$$

$$+\frac{1}{100c(m,n)s(m,n)}\cdot g(i+1,m,n)h(i+1,m,n)$$

$$\leq 100ec(m,n)s(m,n)\left(1+\frac{1}{100c(m,n)s(m,n)}\right)g(i+1,m,n)h(i+1,m,n) \tag{7.12}$$

$$\leq 100ec(m,n)s(m,n)\left(1+\frac{1}{100c(m,n)s(m,n)}\right)(10(f(m,n)+3)^{i+1}h(i+1,m,n)) \tag{7.13}$$

$$\leq 100ec(m,n)s(m,n)\left(1+\frac{1}{100c(m,n)s(m,n)}\right)^{k-i}(10(f(m,n)+3)^{\sum_{j=i+1}^{k} j}h(k,m,n)) \tag{7.14}$$

$$\leq (100ec(m,n)s(m,n))^k(10(f(m,n)+3))^{k^2} \tag{7.15}$$

□

Lemma 3. *Let* $c(m,n), g(m,n), h(m,n)$ *and* $f(m,n)$ *are functions from* $N\times N$ *to* N. *Let* $c(f(m,n)+3)^{c+1} \leq m$ *and* $2(100ec(m,n)s(m,n))^k(10f(m,n)+3)^{k^2+k} \leq n$. *Let* $g(.)$ *and* $h(.)$ *be defined as Lemma 1 and Lemma 2. Then we can get k lists* $L_1,.....,L_k$ *of disjoint sets such that it contains sublists* $L_1,.....,L_k$, *each* L_j *contains g(i, m, n) sets, and each set in* L_i *is of size* $h(i,m,n)$, *and the union of all sets are at most n.*

Proof. We have

$$\sum_{i=1}^{k} g(j,m,n)h(j,m,n) \leq \sum_{i=1}^{k} (100ec(m,n)s(m,n))^k (10(f(m,n)+3))^{k^2} \cdot (f(m+n)+3)^i \quad (7.16)$$

$$\leq 2(100ec(m,n)s(m,n))^k (10(f(m,n)+3))^{k^2} \cdot (f(m+n)+3)^k \quad (7.17)$$

$$\leq 2(100ec(m,n)s(m,n))^k (10(f(m,n)+3))^{k^2+k} \quad (7.18)$$

$$\leq n \quad (7.19)$$

□

Definition 9.

- *If $A_1, ...A_k$ is k list of sets, define union $(A_1, .., .A_k)$ to be the set of union of the sets in the k lists.*
- *For a set S and an integer $t \leq |S|$, let $S[t]$ be defined the first t elements in S.*

Definition 10. *Assume that L is given as Lemma 3 and each L_i has the sets $S_{i,1}, ...S_{i,v_i}$.*

- *Define L_1 to have a list of sublist of sets $L_{1,1}, ..., L_{1,k}$ such that each set $S_{1,j,t}$ in $L_{1,j}$ is $S_{1,1}[h(1,m,n) - |S_{j,t}|] \cup S_{j,t}$ for $t = 1, ..., v_i$.*
- *Let $L_1(j,t)$ is the same as L_1 except that the set $S_{j,t}$ is replaced by $S'_{j,t} = S_{1,1}[h(1,m,n) - |S_{j,t} \cup U_j|] \cup S_{j,t} \cup U_j$, where $U_j = union(L_{1,j}, L_{1,j+1}, ..., L_{1,k})$.*

Lemma 4. *Let L_1 be defined in Definition 10 and $L_2 = L_1(j)$ for some $j \leq k$. Then every set cover solution for L_1 needs at least $\sum_{i=1}^{k} g(i,m,n)$ sets, and an optimal solution for L_2 has $1 + \sum_{i=1}^{j-1} g(i,m,n)$.*

Proof. All sets in L are disjoint to each other. List L has $\sum_{i=1}^{k} g(i,m,n)$ sets. Therefore, every set cover solution for L_1 needs at least $\sum_{i=1}^{k} g(i,m,n)$ sets.

The set $S'_{j,1}$ can cover all elements from $L_{i,j}, L_{1,j+1}, ..., L_{1,k}$. The other part can be covered with $\sum_{i=1}^{j-1} g(i,m,n)$. Thus the optimal solution for L_2 needs $1 + \sum_{i=1}^{j-1} g(i,m,n)$ sets. □

Define two inequalities

$$c(m,n)(f(m,n)+3)^{c(m,n)+1} \leq m, \tag{7.20}$$

$$2(100ec(m,n)s(m,n))^k(10(f(m,n)+3))^{k^2+k} \leq n \tag{7.21}$$

Theorem 6. *Let c(m,n), f(m,n) and s(m,n) be functions $N \times N \to N$. They satisfy the conditions inequalities (7.21) and (7.22) with $k = c(m,n)+1$. Then there is no $c(m,n)$ rounds interactive algorithm for set cover problem with $f(m,n)$-approximation that uses $s(m)$ round samples each phase.*

Proof. Here we construct two list of sets $L_1 : L_{1,1}, ..., L_{1,k}$ with $k = c(m,n)+1$, and $L_2 : L_{2,1}, ..., L_{2,k}$ where $L_{i,j}$ is a sublist of sets in the input list. All sets in the same $L_{i,j}$ have the same size. Let m be the number of sets in the input list. Let function $g(i,m,n)$ and function $h(i,m,n)$ are defined those in Lemmas 1 and 2.

Let L_1 be constructed as that in Definition 10. We construct L_2.

Assume that $T(.,.)$ is an approximation algorithm for set cover problem, and runs in c rounds. Let $T(L,p)$ be the computation of $T(.,.)$ with input L at random path p.

Let $T(L_1,.)$ run on L_1. We have the following claim about some properties of the randomized computation for $T(.)$ with input L_1. □

Definition 11. *Let E_i be the event that $L_{1,i-1}$ has less than $g(i-1,m)$ sets being selected after phase $i-1$, a new elements will be selected from $L(1,j)$ with $j > i-1$ among the $s(m)$ random samples at phase i.*

Claim 4. *Prob*$(E_1 \cup E_2 \cup ..... \cup E_{c(m,n)}) \leq \frac{1}{100}$

Proof. The number of new elements in $L_{1,j}$ with $j > 1$ is at most $\sum_{j=i+1}^{k} h(j,m,n)g(j,m,n) \leq \frac{1}{100c(m,n)s(m,n)} h(i,m,n)g(i,m,n)$ by equation (7.10).

There are $s(m,n)$ random elements. It happens with probability at most $\left(\frac{1}{100c(m,n)s(m,n)}\right)$ $s(m,n) \leq \frac{1}{100c(m,n)}$ at one phase. Therefore, it happens with probability at most $\frac{1}{100}$ at among one of $c(m,n)$ phases with union bound.

Let $\alpha = \frac{1}{2}$, $\beta = \frac{6}{9}$, and $\gamma = \frac{1}{9}$.

Let P_i be the set of all those paths p at stage i that $T(L_1, p)$ takes at least $g(j,m)\gamma$ sets in $L_{1,j}$ with $j \geq i$ at phase i.

Let $\overline{P}_i$ be the complementary set of P_i. Define $U_s[p,i,j] = 1$ if $S \in L_j$, and path p selects S among the first i stages.

We consider two cases at the end of phase i of $T(L_1,.)$:

Case 1. $W(P_i) \geq \alpha$ for some $i \leq c(m,n)$. There is a phase $i \leq c(m,n)$ such that at least total weights α random paths p, $T(L_1,p)$ takes at least $g(j,m)\gamma$ sets in $L_{1,j}$ with $j \geq i$ at phase i. Let i be the least s Let P_i be the set of all those paths p at stage i that $T(L_1,p)$ takes at least $\frac{g(j,m,n)}{9}$ sets in $L_{1,j}$ with $j > i$ at phase i. We consider two cases at the end of phase i of $T(L_1,.)$:

$$\begin{aligned}
\sum_p w(p) \sum_S U_S[p,i-1,i] &= \sum_{p \in P_i} \sum_S U_S[p,i-1,i] + \sum_{p \in \overline{P}_i} w(p) \sum_S U_S[p,i-1,i] \\
&\leq \sum_{p \in P_i} w(p) g(i,m,n) + \sum_{p \in \overline{P}_i} w(p) \gamma g(i,m,n) \\
&\leq g(i,m,n) \sum_{p \in P_i} w(p) + \gamma g(i,m,n) \sum_{p \in \overline{P}_i} w(p) \\
&\leq g(i,m,n)\alpha + \gamma g(i,m,n)(1-\alpha) \\
&\leq g(i,m,n)(\alpha + \gamma(1-\alpha)).
\end{aligned}$$

□

Define $U(S,i,j)$ to be the set of the paths p that selects $S \in L_j$ in the first i phases. Assume

$W(U(S,i-1,i)) \geq \beta$ for all sets S in L_i. Then we have

$$\begin{aligned}\sum_p w(p) \sum_S U_S[p,i-1,i] &= \sum_S \sum_p w(p) U_S[p,i-1,i] \\ &\geq \sum_S \beta \\ &= g(i,m,n)\beta.\end{aligned}$$

Since $\beta \geq (\alpha + \gamma(1-\alpha))$, we have contradiction. Thus, $W(U(S,i-1,i) \leq \beta$ for some set S in L_i.

At phase $i-1$, there is a $S_{i,t}$ that is not picked up by at least $\frac{2}{9}$ portion of paths. Let $L_2 = L_1(i+1,t)$.

We will find that L_1 and L_2 have greatly different solution for the set cover problem. On the other hand, they will have the same output. This will make that the number of sets for two set cover problems have big difference. This brings a contradiction by Lemma 4. This is because $\gamma \cdot g(i+1,m,n) \geq f(m,n)(1+\sum_{j=1}^{i} g(j,m,n))$ by equations (7.1) and (7.2). For the input L_2, the algorithm selects at least $\gamma g(i+1,m,n)$ sets for most of the paths, but it only needs at most $(1+\sum_{j=1}^{i} g(j,m,n))$ sets to cover all elements. For inputs L_1 and L_2, the algorithm gives the same output for most of the paths. A set cover solution for L_1 is not a $f(m,n)$-solution for L_2 by equation (7.2).

Case 2: $W(P_i) < \frac{1}{2}$ for all $i \leq c(m,n)$. For each $i \leq c(m,n)$. In particular, we have $W(P_{c(m,n)}) < \frac{1}{2}$. Thus, $W(\overline{P}_{c(m,n)}) = 1 - W(P_{c(m,n)}) > \frac{1}{2}$. For each paths $p \in \overline{P}_{c(m,n)}$, $T(L_1,p)$ picks less than $\gamma g(j,m,n)$ sets in $L_{1,j}$ with $j = c(m,n)+1$.

We do not have enough sets from $L_{1,k}$ with $k = c(m,n)+1$ to cover the elements in the sets of $L_{c(m,n)+1}$. In this case, the algorithm with input L_1 fails at all the paths in $P_{c(m,n)}$ with $W(\overline{P}_{c(m,n)}) > \frac{1}{2}$ as the sets selected among those paths in $P_{c(m,n)}$ cannot cover all elements in the universal set. Note that phase $c(m,n)$ is the final phase of the algorithm. No more sets will be

picked up after phase $c(m,n)$. Thus, we have contradiction that the randomized algorithm returns a set cover among paths with sum of weights at least $\frac{3}{4}$.

Corollary 1: For each $d \geq 0$, there is a positive d_1 such that there is no $\left(d_1\sqrt{\frac{logn}{loglogn}}), m^d\right)$ interactive algorithm for the set cover problem with $dlog(n)$ approximation ratio.

Proof. Assume that the number of random samples is m^d, and $f(m,n) = dlog(n)$. We can satisfy inequalities (7.21) and (7.22) with $k = c(m,n)+1$ in the condition $c(m,n) = d_1\sqrt{\frac{logn}{loglogn}}$ for some fixed d_1. It follows from Theorem 6. □

Corollary 2: For a fixed $d \geq 0$, there is no (d, m^d) interactive algorithm for the set cover problem with n^α approximation ratio for some fixed $\alpha \geq 0$.

Proof. Assume that the number of random samples is m^d, and $f(m,n) = n^\alpha$. We can satisfy inequalities (7.21) and (7.22) with $k = c(m,n)+1$ in the condition for some fixed $\alpha \geq 0$. It Follows from Theorem 6. □

Corollary 3: For a fixed $d \geq 0$, there is no $(d, n^{1-\varepsilon})$ interactive algorithm for the set cover problem with n^α approximation ratio for some fixed $\alpha \geq 0$.

Proof. Assume that the number of random samples is $n^{1-\varepsilon}$, and $f(m,n) = n^\alpha$. We can satisfy inequalities (7.21) and (7.22) with $k = c(m,n)+1$ in the condition for some fixed $\alpha \geq 0$. It Follows from Theorem 6. □

CHAPTER VIII

CONCLUSION

In this paper, we develop a randomized interactive model for solving set cover problem. It needs $\mathcal{O}(\log n)$ phases and each phase takes $poly(m)$ random samples from those uncovered elements. An interesting problem is to close the gap between the $\mathcal{O}(\log n)$ upper bound and $\Omega\left(\sqrt{\frac{logn}{loglogn}}\right)$ lower bound for the number of phases under the condition that each of the phase uses $poly(m)$ random samples from uncovered elements.

BIBLIOGRAPHY

[1] A. CHAKRABARTI AND A. WIRTH, *Incidence geometries and the pass complexity of semi-streaming set cover*, in Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, Society for Industrial and Applied Mathematics, 2016, pp. 1365–1373.

[2] V. CHVATAL, *A greedy heuristic for the set-covering problem*, MATHEMATICS OF OPERATIONS RESEARCH, 4 (1979).

[3] G. CORMODE, H. KARLOFF, AND A. WIRTH, *Set cover algorithms for very large datasets*, in Proceedings of the 19th ACM international conference on Information and knowledge management, ACM, 2010, pp. 479–488.

[4] E. D. DEMAINE, P. INDYK, S. MAHABADI, AND A. VAKILIAN, *On streaming and communication complexity of the set cover problem*, in International Symposium on Distributed Computing, Springer, 2014, pp. 484–498.

[5] Y. EMEK AND A. ROSÉN, *Semi-streaming set cover*, ACM Transactions on Algorithms (TALG), 13 (2016), p. 6.

[6] U. FEIGE, *A threshold of ln n for approximating set cover*, Journal of the ACM (JACM), 45 (1998), pp. 634–652.

[7] T. A. FEO AND M. G. RESENDE, *A probabilistic heuristic for a computationally difficult set covering problem*, Operations research letters, 8 (1989), pp. 67–71.

[8] S. GOLDWASSER, S. MICALI, AND C. RACKOFF, *The knowledge complexity of interactive proof systems*, SIAM Journal on computing, 18 (1989), pp. 186–208.

[9] W. HOEFFDING, *Probability inequalities for sums of bounded random variables*, Journal of the American statistical association, 58 (1963), pp. 13–30.

[10] I. INTRODUCING, *Interactive proof systems*, (2006).

[11] D. S. JOHNSON, *Approximation algorithms for combinatorial problems*, Journal of computer and system sciences, 9 (1974), pp. 256–278.

[12] R. M. KARP, *Reducibility among combinatorial problems*, in Complexity of computer computations, Springer, 1972, pp. 85–103.

[13] R. KUMAR, B. MOSELEY, S. VASSILVITSKII, AND A. VATTANI, *Fast greedy algorithms in mapreduce and streaming*, ACM Transactions on Parallel Computing, 2 (2015), p. 14.

[14] L. LOVÁSZ, *On the ratio of optimal integral and fractional covers*, Discrete mathematics, 13 (1975), pp. 383–390.

[15] R. MOTWANI AND P. RAGHAVAN, *Randomized algorithms*, Chapman & Hall/CRC, 2010.

[16] B. SAHA AND L. GETOOR, *On maximum coverage in the streaming model & application to multi-topic blog-watch*, in Proceedings of the 2009 SIAM International Conference on Data Mining, SIAM, 2009, pp. 697–708.

[17] S. K. STEIN, *Two combinatorial covering theorems*, Journal of Combinatorial Theory, Series A, 16 (1974), pp. 391–397.

[18] N. E. YOUNG, *Greedy set-cover algorithms*, in Encyclopedia of algorithms, Springer, 2008, pp. 379–381.

APPENDIX A

APPENDIX A

CHERNOFF BOUND

Proof. Chernoff bound is a tail bound inequality which bounds the amount of probability of some random variable S that is far from the mean.

Let $t > 0$. From the definition of the expectation, we have $E(e^{tS_i} = Pr(S_i = 1)e^t + Pr(S = 0)$. For the proof of this inequality, we first consider a function $f(x) = e^x - (1+x)$. The function is always positive at $f'(x)$ for $x > 0$ and negative for $x < 0$. Let X_i be a 0-1 random variable for$i = 1...n$ with expectation p_i. We have,

$Pr[S \geq (1+\delta)\mu] = Pr[e^{tS} \geq e^{t(1+\delta)\mu}]$

Now we apply the Markovs inequality in above equation, we get

$$Pr[S \geq (1+\delta)\mu] = Pr[e^{tS} \geq e^{t(1+\delta)\mu}] \tag{1.1}$$

$$\leq \frac{E[e^{tS}]}{e^{t(1+\delta)\mu}} \tag{1.2}$$

$$\leq \frac{e^{e^t-1}\mu}{e^{t(1+\delta)\mu}} \tag{1.3}$$

$$= \left(\frac{e^{e^t-1}}{e^{t(1+\delta)}}\right)^{\mu} \tag{1.4}$$

$$= (e^{e^t-1-t(1+\delta)})^{\mu}. \tag{1.5}$$

□

Now we choose t in such a way that exponent $e^t - 1 - t(1+\delta)$ gets minimized. Applying derivative

with respect to t to 0 gives $e^t = (1+\delta)$ or $t = ln(1+\delta)$. Plugging this value gives

$$Pr[S \geq (1+\delta)\mu] \leq \left(e^{(1+\delta)-1-(1+\delta)ln(1+\delta)}\right)^{\mu} \tag{1.6}$$

$$= \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta)}}\right)^{\mu} \tag{1.7}$$

BIOGRAPHICAL SKETCH

Ujjwol Subedi was born in a small village called Sarlahi in eastern part of Nepal to his parents Kalpana Subedi and Lok Nath Subedi Sharma. He obtained his high school degree from National School of Sciences. He finished his undergraduate from North Dakota State University in 2013 with Bachelor's Degree in Computer Science and Mathematics. After completion of degree, he worked as a system analyst in Softnice Inc from 2014 - 2015. After working one year he decided to pursue his Master Degree from University of Texas Rio Grande Valley where he received his Master of Science in Computer Science degree in May 2017.

Permanent Address:
Ujjwol Subedi
Dhumbarahi - ward no 4
Kathmandu, Nepal 00977
ujjwol007@gmail.com