2019

Variation In Greedy Approach To Set Covering Problem

Shreeya Naval Singhania
University of Windsor

VARIATIONS IN GREEDY APPROACH TO SET COVERING PROBLEM

By

Shreeya Singhania

A Thesis
Submitted to the Faculty of Graduate Studies
through the School of Computer Science
in Partial Fulfillment of the Requirements for
the Degree of Master of Science
at the University of Windsor

Windsor, Ontario, Canada

2019

VARIATIONS IN GREEDY APPROACH TO SET COVERING PROBLEM

by

Shreeya Singhania

APPROVED BY:

H. Wu

Department of Electrical and Computer Engineering

J. Lu

School of Computer Science

J. Chen, Advisor

School of Computer Science

April 8, 2019

DECLARATION OF ORIGINALITY

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication.

I certify that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances to my appendix.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

ABSTRACT

The weighted set covering problem is to choose a number of subsets to cover all the elements in a universal set at the lowest cost. It is a well-studied classical problem with applications in various fields like machine learning, planning, information retrieval, facility allocation, etc. Deep web crawling refers to the process of gathering documents that have been structured into a data source and can be retrieved through a search interface. Its query selection process calls for an efficient solution to the set covering problem.

Within this context, the data follows the lognormal and power law distribution, and a TS-IDS algorithm has been proposed in the literature and demonstrated to outperform both the greedy and IDS algorithm. We have evaluated the performance of various greedy approaches to the set covering problem, including the TS-IDS, using open source dataset in the context of resource management. The data are sampled from a given roadmap with different coverage radius.

DEDICATION

Dedicated to my parents Naval Singhania and Manju Singhania

and also rest of my family and friends

ACKNOWLEDGEMENTS

I would like to thank my thesis advisor Dr. Jessica Chen for her valuable guidance and continuous support during my research. She always guided me in the right direction, and this work would not have been accomplished without her help. I would also like to thank my thesis committee members Dr. Huapeng Wu and Dr. Jianguo Lu for their valuable comments and professional assistance.

I would also like to thank Karen Bourdeau and the entire faculty of School of Computer Science for their continuous support throughout the program. I would like to express my profound gratitude to my parents and family members who were the motivation and strength behind this work. Finally, I want to thank every one of my friends who remained by amid great and awful circumstances.

Shreeya Singhania

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1 Overview

One of the common problems faced by the people in today's world is resource planning. Planning not only includes thinking about where to construct new places in a particular city, how to provide the resources to the people, but also includes how the resource can be optimally used. The answer to this problem comes with many techniques and out of all, one method can be the solution to the set covering problem. Searching for the smallest set of solution from a large set of solutions, which is extensively studied in many fields.

1.1.1 Set Covering Problem

Set Covering is a common problem in the field of computer science. It is one of Karp's 21 NP-complete problem. Set covering is examined NP-Hard in optimization and search problems, NP-Complete in the decision-based problem.

The problem related to Set Cover is a combinatorial optimization problem, which is seen in the real world context, where we have a collection of needs (eg., tasks, responsibilities) and a collection of resources (eg., employees or machines) and the task is to find a minimal set of resources to satisfy the need.

Set covering means to choose a minimum number of subsets in such a way that they cover all the elements from a universal set. The main factor in the process is to reduce the cost of the chosen subset.

There are various applications of set covering.

Some of them are as follows [1]:

1. In operational research, we need to place the services so that it can be reached out by all the sites that are close to the facility. This is termed as Set Cover. Here, the base set is the set of all sites to be covered by the service locations.
2. In planning, it is required to select how to place the resources in a city when the demands are dynamically changing. An alternative approach of Set Covering Problem is used to cover this entire requirement [3].
3. In terms of data quality, a set of simple rules could be used to illustrate the observed data in order to help users understand the structure in their data. Given a collection of rules that are consistent with the observed data, the tableau generation problem is to find a subset of rules which explains the data without redundancy. This is captured by applying Set Cover to the collection of rules [4].
4. For information retrieval, a query is used to retrieve the smallest set of documents to cover a set of topics i.e., to find a set cover [16].

Set covering problem is NP-Hard. There are typically two ways to solve this problem: exact and heuristic approaches. The exact algorithms are those that can give the optimal solution to an optimization problem. But in combinatorial problems, conventional methods are not efficient enough, especially when the problem is large and complex, including those based on branch and bound, branch and cut [5]. The best exact algorithm for the SCP is the one adopted in software package CPLEX and MINTO by [17]. A heuristic approach means a set of steps that do not guarantee an optimal solution. An approximation algorithm is the one that leads us reasonably close to an optimal solution.

1.2 Resource Management

Before managing the resources, one should know what it means. It means managing of resources so that they can be provided effectively and efficiently when they are needed. Such resources can be human resource, financial resource, inventory, human skills or even Information Technology (IT), etc. We need to know the data, which will be required for effective resource management, including demand for it, availability of resources over the period of time, and how these resources will fit into the demand. As we know, resources are basically used to minimize the cost by using at maximum request. Resources should be allocated in a fair and balanced way.

There are many key factors of resource management. Some examples are: -

- Resource Plan: Everyone needs to plan for how to use it, for example, in the project plan, we need to consider resource plan as a key component, as it will contain the entire planning from the start till the end.
- Resource Breakdown Structure: In breakdown structure, we breakdown the plan into some hierarchy to complete a particular project easily and effectively.

1.3 Motivation

In today's world, we need resources for completing any specific task, whether it is by a human being or by machine. The main problem is the managing of resources, which needs to be studied in order to get the best use of resources for an extended period of time. In the field of planning, we need to think about the management of the resource, where various related problems need to be addressed.

In big firms like public or private sector, facility location is one of the critical components for planning. Facility Location is a well-established part of Operational Research in itself [7]. For the decision-making process, there are various models that can help, and out of all popular models, one is the facility location model in covering problem. The problem here is to determine both the number and the location of public facilities like schools, libraries, parks, fire station, etc.

Another main reason for studying in this area is the need to properly plan things. Due to the fast growth of population, considering some resources has pushed us to think about finding a solution, in order to provide it to people in some way that demands can be completed in the best way. One real life example could be placing of the hospital in a city in such a way that, hospitals can be easily reached by the entire area of the city at a minimum cost.

One existing solution to SC problem is the greedy approach, which follows the problem-solving heuristic, always making the choice of the movement that seems to be best and it always selects the set with a large number of uncovered items repeatedly.

The TS-IDS algorithm by Wang et.al [8] has proven to be a better variation of greedy approach for query processing and it has also outperformed both IDS and greedy approach but hasn’t been applied to different applications. The testing of TS-IDS algorithm in the field of the resource management may give some good results.

1.4 Problem Statement

As the world is facing the problem of shortage of resources, there comes a need where it is required to either manage or get resources appropriately. Research on Set Covering has given many great results for planning and resource management problems.

A TS-IDS Algorithm for the set covering problem has been proposed by [8] within the context of web crawling and demonstrated to outperform the traditional greedy algorithm and IDS algorithm. In our present work, we measure the performance of the various greedy approaches on open-source datasets in the context of resource management. It includes the setting of resource allocation on a given road map with 1.9 million nodes. The dataset is taken from open source data provided by Stanford Network Analysis Project (SNAP).

1.5 Contribution

In our thesis, the contribution is we have tested different greedy approaches on the 5000 by 5000 sample data where each dataset is of different coverage radius in total 50 sample data using SNAP dataset. We have also used two different cost definitions such as location and infrastructure cost where we have run our different greedy approaches ten times on each 50 sample dataset and last is the data distribution where we have calculated coefficient of variance for each dataset and compared with the improvement of it.

The rest of this thesis is organized as follows: In Chapter 2, we define the problem and present the proposed approach. Chapter 3 provides a review of some of the concepts and terminologies that are related to this work and provides more details of the areas related to this research. It also includes a review of some of the closely related work of other researchers. Chapter 4 shows the results of experiments and Chapter 5 concludes the thesis with some suggested future work.

CHAPTER 2

BACKGROUND

In this chapter we introduce Coefficient of variance, which was used in TS-IDS algorithm [8] and greedy algorithm in the field of resource management. As seen in the paper, the TS-IDS has outperformed both greedy and weighted greedy algorithm in the field of web crawling. Our objective is to compare various greedy approaches on the roadmap dataset.

2.1 Greedy Algorithm for Set Covering Problem:

There are cases where the greedy algorithm (GA) results in an optimal solution. But, in many instances, this may not be achieved. For example, the set cover problem states the greedy algorithm may not result in an optimal solution. As we know already, the greedy algorithm does provide the "best" solution at each step. As it is said, a choice that is locally best doesn't mean to be a globally best choice. In paper [21], [22], greedy algorithms are well known polynomial time approximation algorithm for set cover. Because of their easy implementation, they are commonly used the heuristic algorithm.

In SCP, we know we are given a set of universal elements U such that |U| = n, and sets S1... Sk ⊆ U. A set cover is a collection C of some of the sets from S1…, Sk whose union is the entire universe U. Formally, C is a set cover if $U_{s_i \in c} S_i = U$. We want to minimize cost |C|.

Greedy Set Cover (M, N)

```
1. U <- M
2. X <- $\phi$
3. While U $\neq \phi$ do
        a. select an S ∈ N that maximizes $|S \cap U|$
        b. U <- U - S
        c. X <- X ∪{S}
```

4. return X

At each stage, the greedy approach will select the set Si that contains most uncovered elements in it. This step is repeated until all the elements in the set are covered. Let M = {1,2,3,4,5,6,7,8,9,10} be a universal set and S1= {1, 2, 3, 4, 5, 6}, S2= {1, 2, 3, 7, 8}, S3= {4,5,6,9,10}, S4= {7, 8, 9} and S5= {10}. Initially C is empty.

In the first step, the greedy algorithm selects the set S1= {1,2,3,4,5,6} because it has better coverage than other subsets. The solution set is C = {S1}.

In the next step, S4 has most uncovered elements {7,8,9}, hence the greedy will now select this set. Now the solution set is C= {S1, S2}.

In the third step, S2 and S5 have only one uncovered element {10} and {10}. Any one of them will be selected. Let us select S5, so the solution set will be C= {S1, S2, S5}. Hence, all the elements of M set are covered. So C is the final solution.

	1	2	3	4	5
1	0	1	0	1	0
2	0	1	1	1	0
3	0	0	1	1	0
4	1	1	1	1	1
5	1	0	0	0	1
nd	2	3	2	5	2
$\boldsymbol{nw}$	2	3	3	4	2
$\boldsymbol{nw/nd}$	1	1	1.5	0.8	1

Fig 2.1: Sample matrix for the greedy algorithm

The above figure shows an example matrix used in greedy algorithm, where nd stands for node degree, i.e., the number of nodes getting covered by a node, nw stands for new node frequency, i.e., the number of new nodes getting fetched by a particular node, nw/nd is used to determine which node should be the next solution node.

2.2 TS-IDS algorithm

In TS-IDS algorithm, the TS (term size) is the number of documents it can cover, i.e., document frequency, and IDS (inverse of document size) is used to define the document weight. It is assumed that the document importance depends on the sizes of the term, i.e., the number of documents it can cover, and on the number of terms it contains. The reason is that the large term will bring in a greater number of duplicates Therefore, we define the document importance to be proportional to the minimal term size in it.

Definition: the document weight is defined as $d = \frac{1}{d_i} \times \min(TS)$. The node with maximum μ / nd will be selected. Here $\mu = \min(TS)$. For example, in the following table node 4 will get selected as it has maximum value.

	1	2	3	4	5
1	0	1.5	0	1.5	0
2	0	1	1	1	0
3	0	0	1.5	1.5	0
4	0.4	0.4	0.4	0.4	0.4
5	1	0	0	0	1

Nd	2	3	3	4	2
nw	1.4	2.9	2.9	4.9	1.4
nw/nd	0.7	0.966	1.133	1.225	0.7

Fig 2.2: TS-IDS algorithm

2.3 Roadmap

Here roadmap is defined as a graph of roads containing the edge and node for connecting the particular areas. Each node represents an intersection, and each edge represents a road between two nodes. A matrix can be used to represent such a graph of nodes with 0-1 value denoting which node is connected to which another node.

2.4 Setup Process

The main purpose of our experiment is to test the performance of various greedy approaches in the field of resource management and to compare them.

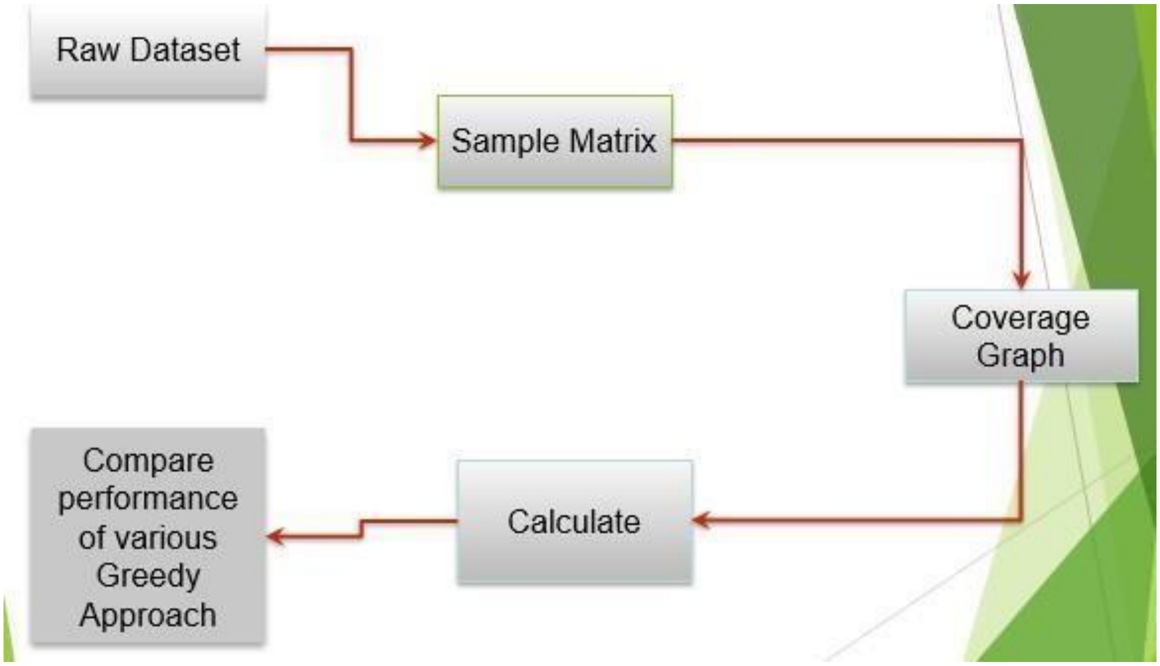


Fig 2.3: Step by step process of the approach

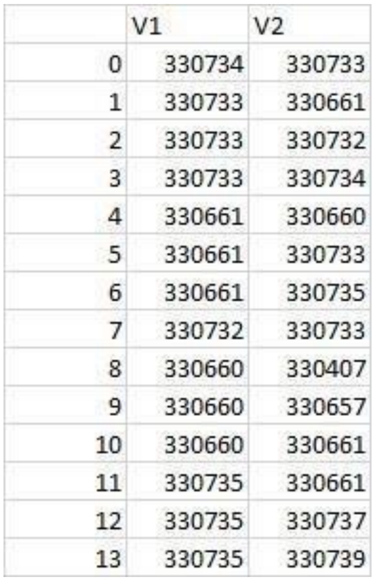
The first three steps are the part of the preprocessing where the data is taken from the raw dataset and converted into the subgraph, and a coverage matrix is created and then passed as the required input to the various greedy approaches. The next step is to calculate the coefficient of variance which is a major factor for comparing the approaches. The last step is the comparison of various greedy approaches, where all approaches will be implemented and run with the same input to test their performance.

2.4.1 Raw Dataset

The raw dataset is the actual data taken from the SNAP (Stanford Network Analysis Project) [48]. The dataset we adopted is a roadmap network of California. The map contains 1,965,206 nodes and 2,766,607 edges.

2.4.2 Subgraph

It is always a difficult task when it comes to working with a large dataset due to some problems such as memory usage and hardware resources. In order to avoid these problems, we can use subgraph, which is obtained from the original graph. A subgraph is a regional network of roads formed from the original graph.



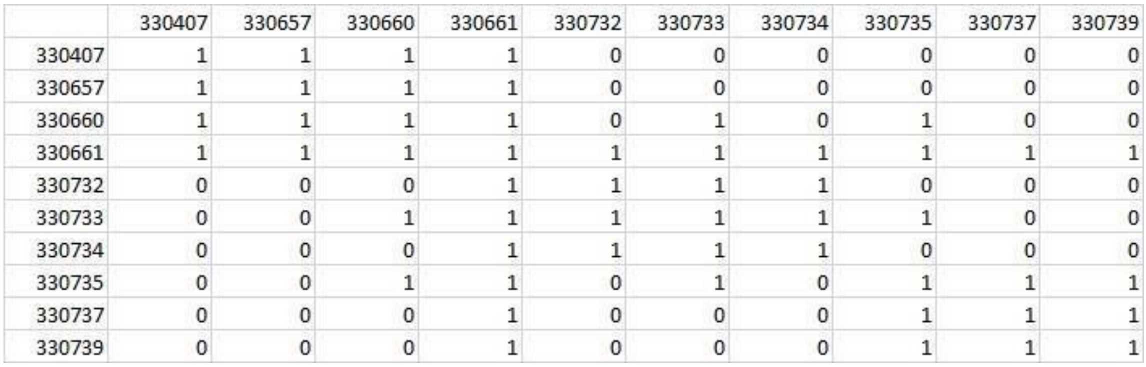
	V1	V2
0	330734	330733
1	330733	330661
2	330733	330732
3	330733	330734
4	330661	330660
5	330661	330733
6	330661	330735
7	330732	330733
8	330660	330407
9	330660	330657
10	330660	330661
11	330735	330661
12	330735	330737
13	330735	330739

Fig 2.4: Subgraph of 10 nodes

The above figure is a subgraph of 10 nodes connected to each other, taken from the raw dataset. We are going to use this type of subgraph as input to create a coverage matrix.

2.4.3 Coverage Matrix

Coverage matrix is created from the connected subgraph. It is created with a defined coverage radius. Here, coverage refers to the nodes that are getting covered starting from a particular node. As the coverage radius increases, a greater number of nodes will get covered. For example, if radius is two for the matrix, then we say a node can cover its neighbours and neighbours of their neighbours. The information is represented in the form of an adjacency 0-1 matrix. If the node gets covered it is marked as 1, otherwise 0.



	330407	330657	330660	330661	330732	330733	330734	330735	330737	330739
330407	1	1	1	1	0	0	0	0	0	0
330657	1	1	1	1	0	0	0	0	0	0
330660	1	1	1	1	0	1	0	1	0	0
330661	1	1	1	1	1	1	1	1	1	1
330732	0	0	0	1	1	1	1	0	0	0
330733	0	0	1	1	1	1	1	1	0	0
330734	0	0	0	1	1	1	1	0	0	0
330735	0	0	1	1	0	1	0	1	1	1
330737	0	0	0	1	0	0	0	1	1	1
330739	0	0	0	1	0	0	0	1	1	1

Fig 2.5: Coverage matrix for coverage value 2

The above figure represents the subgraph in the form of a coverage matrix, where each row shows what nodes it can cover. For example, in the above matrix, the radius is two so, each node can cover its own neighbours and neighbours of their neighbours. In row 2, for example, 330657 can cover column 330660 and then can cover 330407 because it is a neighbor of 330660 and so on. The coverage matrix will now be the input for various greedy approaches.

2.5 Coefficient of Variation

A coefficient of variation (CV) can be defined as the measurement of relative variability. It is the ratio of the standard deviation σ to the mean μ. It is most commonly used in the field of physics and engineering for quality assurance studies and by economist and investor for modelling economics and estimating the volatility of a security. The main advantage of using CV is that it is unit less. It is also used for presenting the consistency of the data. Consistency refers to the uniformity of data distribution. It is also used to compare two datasets or samples. The coefficient of variation can be defined as:

$$CV = \frac{S}{x}$$

CV can only be used for computing data measured on a ratio scale and only for non-negative values. It can only be used for data on the interval scale. In the above formula, x stands for the mean of the sample data, and s as standard deviation.

The disadvantage of the Coefficient of Variation is that if the mean is close to zero it will reach infinity.

In our experiment, for each sample, we calculate the CV. Once the coefficient of variance is calculated, we use the specific range of CV to analyze the improvement our algorithm shows.

In our approach, we use the Coefficient of Variance formula in the setting of [34]. The dispersion of a node degree (deg) is measured using CV. According to the CV formula, mean is defined by the sum of node degrees over the total number of nodes in the sample

$$\mu = \frac{\sum_{i=1}^{m} \deg(i)}{m}$$

The mean is also known as average node degree. The standard deviation (SD) is defined as

$$SD = \sqrt{\frac{\sum_{i=1}^{m} (\deg(i) - \mu)^2}{m}}$$

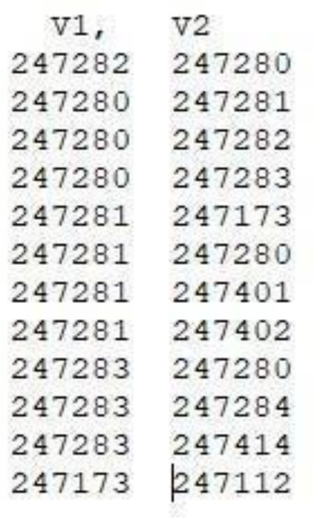
where, m is the total number of nodes in the sample, μ is the calculated mean and deg is the degree of a node. From the above formula, we calculate the CV as follows:

$$CV = \frac{1}{\mu}\sqrt{\frac{\sum_{i=1}^{m} (\deg(i) - \mu)^2}{m}}$$

Now we can use the above formula for comparing various greedy approaches on data with specific CV.

2.6 Example

An illustrative example is given on a small dataset. Here the dataset is of ten nodes and all steps are performed on it. Assume that we have the following subgraph from the raw dataset



```
  v1,    v2
247282 247280
247280 247281
247280 247282
247280 247283
247281 247173
247281 247280
247281 247401
247281 247402
247283 247280
247283 247284
247283 247414
247173 247112
```

Fig 2.6: Subgraph of 10 nodes

The plot view diagram of the subgraph is :

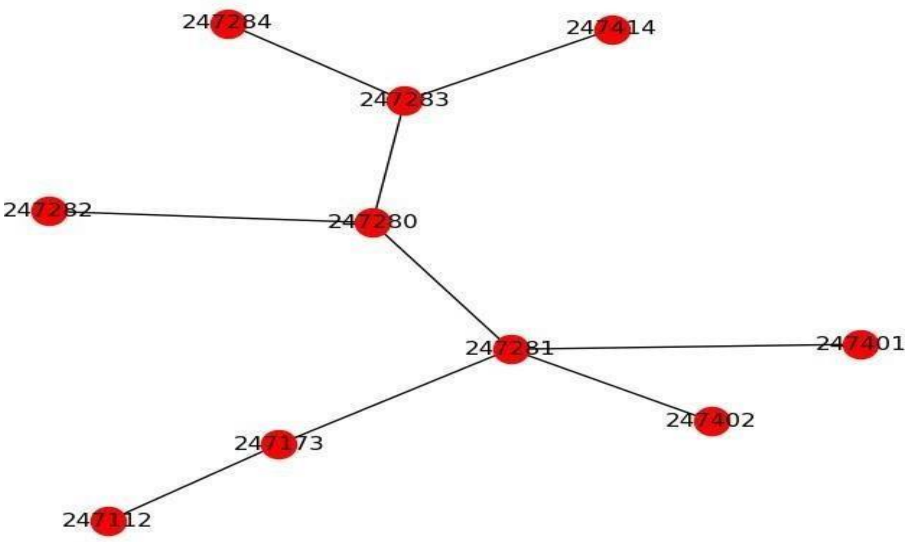
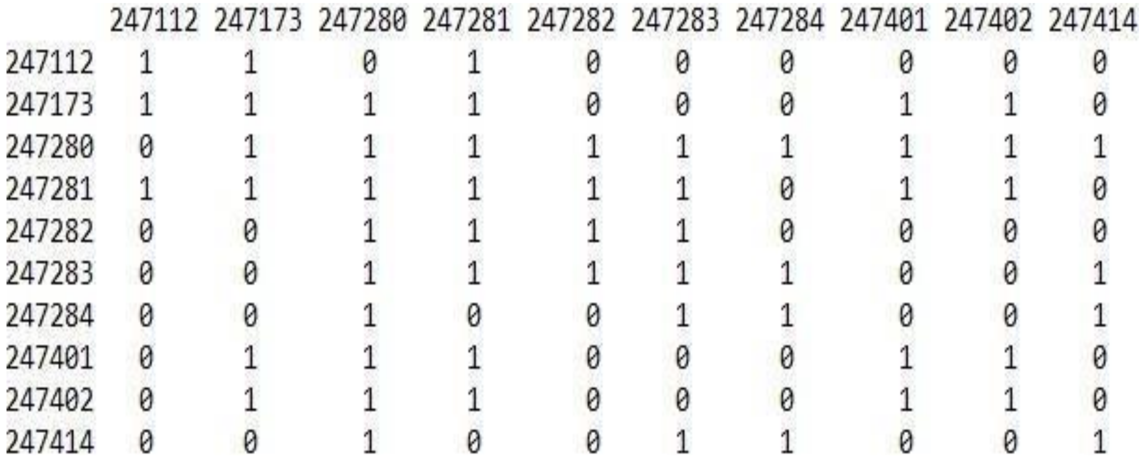


Fig 2.7: Connected subgraph of the 10 nodes

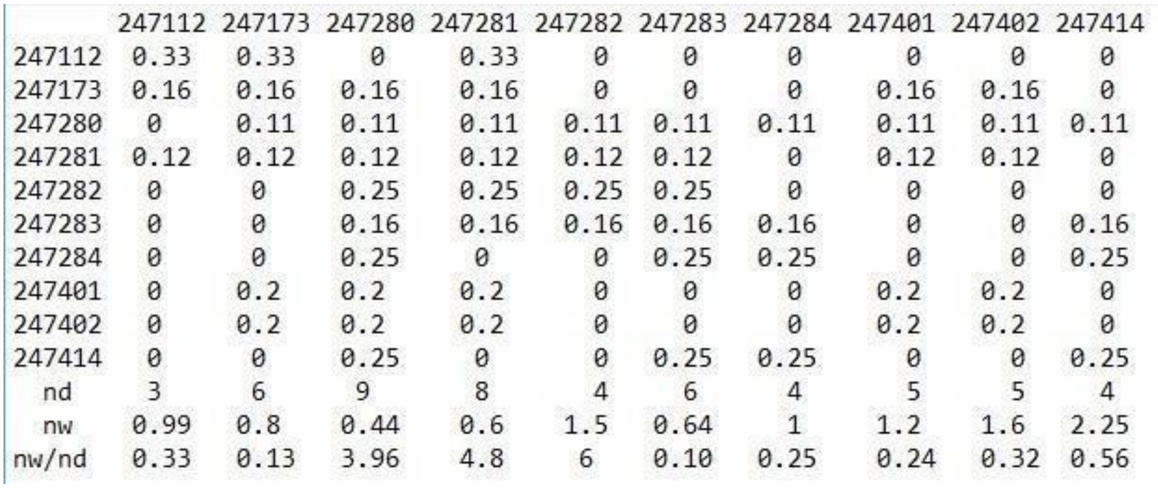
Now the above subgraph is converted into 0-1 matrix with coverage radius 2.



	247112	247173	247280	247281	247282	247283	247284	247401	247402	247414
247112	1	1	0	1	0	0	0	0	0	0
247173	1	1	1	1	0	0	0	1	1	0
247280	0	1	1	1	1	1	1	1	1	1
247281	1	1	1	1	1	1	0	1	1	0
247282	0	0	1	1	1	1	0	0	0	0
247283	0	0	1	1	1	1	1	0	0	1
247284	0	0	1	0	0	1	1	0	0	1
247401	0	1	1	1	0	0	0	1	1	0
247402	0	1	1	1	0	0	0	1	1	0
247414	0	0	1	0	0	1	1	0	0	1

Table 2.1:0-1 matrix

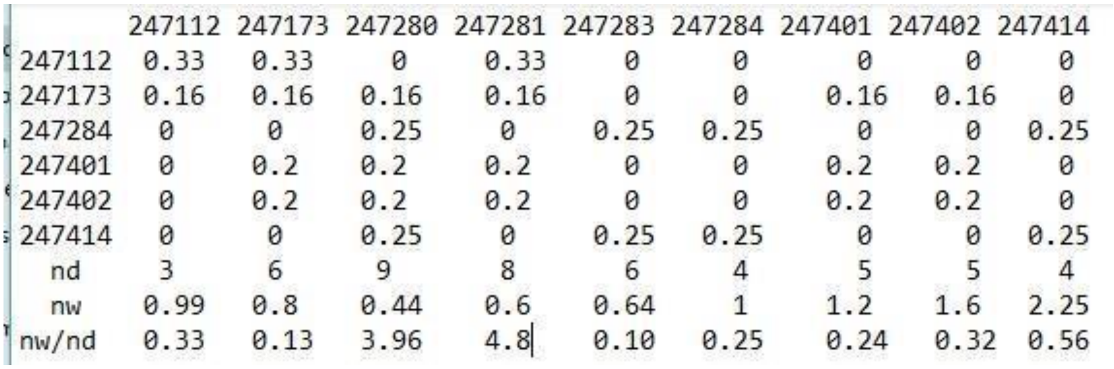
Now we apply the TS-IDS algorithm on the above matrix and calculate the node degree (nd), node weight (nw), and nw/nd. First, we sum up all the values of each row then replace each 1's in this row with the inverse of this sum. Then, we multiple each value with the minimum $column$ value. The same will be done for all the rows and the resulting matrix will be:



	247112	247173	247280	247281	247282	247283	247284	247401	247402	247414
247112	0.33	0.33	0	0.33	0	0	0	0	0	0
247173	0.16	0.16	0.16	0.16	0	0	0	0.16	0.16	0
247280	0	0.11	0.11	0.11	0.11	0.11	0.11	0.11	0.11	0.11
247281	0.12	0.12	0.12	0.12	0.12	0.12	0	0.12	0.12	0
247282	0	0	0.25	0.25	0.25	0.25	0	0	0	0
247283	0	0	0.16	0.16	0.16	0.16	0.16	0	0	0.16
247284	0	0	0.25	0	0	0.25	0.25	0	0	0.25
247401	0	0.2	0.2	0.2	0	0	0	0.2	0.2	0
247402	0	0.2	0.2	0.2	0	0	0	0.2	0.2	0
247414	0	0	0.25	0	0	0.25	0.25	0	0	0.25
nd	3	6	9	8	4	6	4	5	5	4
nw	0.99	0.8	0.44	0.6	1.5	0.64	1	1.2	1.6	2.25
nw/nd	0.33	0.13	3.96	4.8	6	0.10	0.25	0.24	0.32	0.56

Table 2.2: Matrix after calculating nf, nw, nf/nw

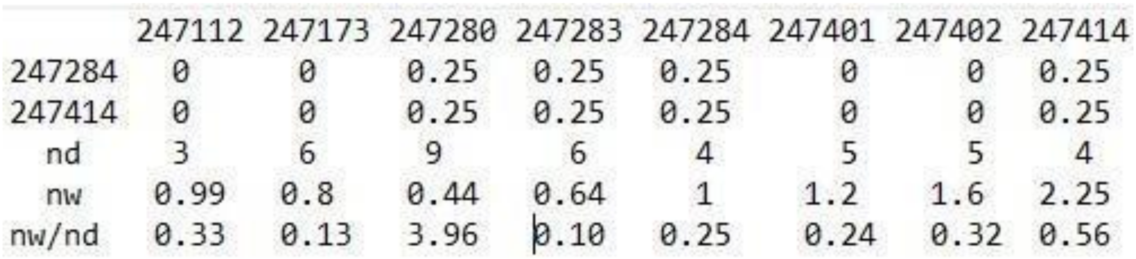
Now we select the node with the maximum nf/nw value. From the above column 247282 is selected as it has the maximum value of 6. All rows covered by this row are then removed. This include column 247282. The resulting matrix after removing rows and columns will be:



	247112	247173	247280	247281	247283	247284	247401	247402	247414
247112	0.33	0.33	0	0.33	0	0	0	0	0
247173	0.16	0.16	0.16	0.16	0	0	0.16	0.16	0
247284	0	0	0.25	0	0.25	0.25	0	0	0.25
247401	0	0.2	0.2	0.2	0	0	0.2	0.2	0
247402	0	0.2	0.2	0.2	0	0	0.2	0.2	0
247414	0	0	0.25	0	0.25	0.25	0	0	0.25
nd	3	6	9	8	6	4	5	5	4
nw	0.99	0.8	0.44	0.6	0.64	1	1.2	1.6	2.25
nw/nd	0.33	0.13	3.96	4.8	0.10	0.25	0.24	0.32	0.56

Table 2.3: Matrix after removing the rows covered by the first selected column

Again the same process gets repeated. The next maximum value will get selected. This time, we select column 247281 and rows covered by this node will be removed. Then again the matrix will be updated.



	247112	247173	247280	247283	247284	247401	247402	247414
247284	0	0	0.25	0.25	0.25	0	0	0.25
247414	0	0	0.25	0.25	0.25	0	0	0.25
nd	3	6	9	6	4	5	5	4
nw	0.99	0.8	0.44	0.64	1	1.2	1.6	2.25
nw/nd	0.33	0.13	3.96	0.10	0.25	0.24	0.32	0.56

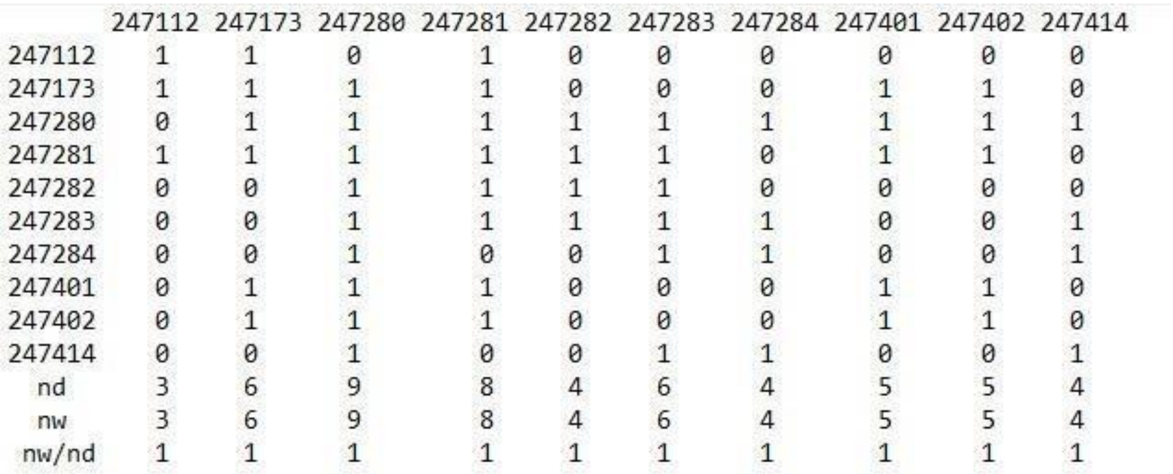
Table 2.4: matrix after removing 2nd column

Now in the 3rd round, column 247280 gets selected and rows covered by this node are removed. The solution contains nodes selected that are 247282,247281,247280

node	***nd***	***Nw***	***Nw/nd***	***cost***	***Uniquerows***
247282	4	0.6	0.15	4	4
247281	8	0.6	4.8	8	8
247280	9	0.44	3.96	10	11

Table 2.5: Results for TS-IDS Algorithm

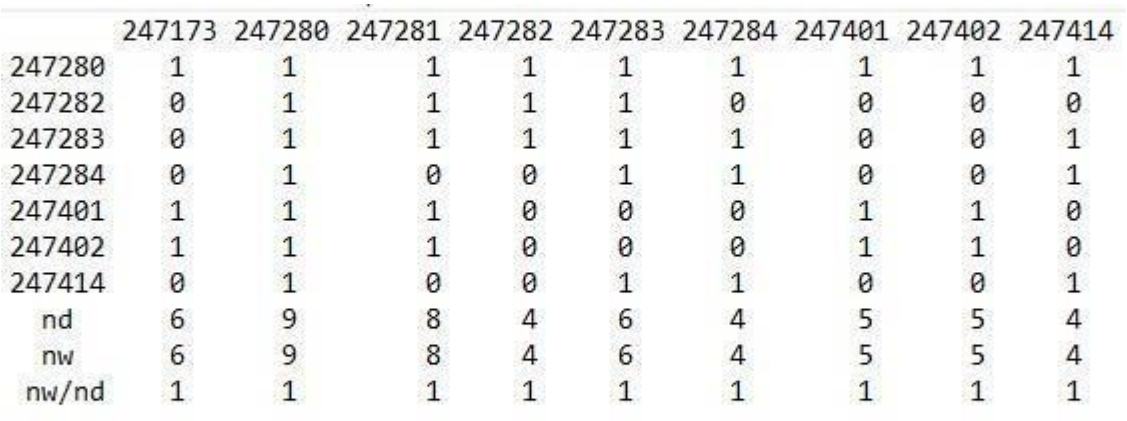
In the next example, a greedy algorithm will be performed on this 10 node sample. This algorithm will randomly select any node at the beginning because every node is initially marked as 1, and the final answer of nw/nd is 1. Let's say it starts with 247112.



	247112	247173	247280	247281	247282	247283	247284	247401	247402	247414
247112	1	1	0	1	0	0	0	0	0	0
247173	1	1	1	1	0	0	0	1	1	0
247280	0	1	1	1	1	1	1	1	1	1
247281	1	1	1	1	1	1	0	1	1	0
247282	0	0	1	1	1	1	0	0	0	0
247283	0	0	1	1	1	1	1	0	0	1
247284	0	0	1	0	0	1	1	0	0	1
247401	0	1	1	1	0	0	0	1	1	0
247402	0	1	1	1	0	0	0	1	1	0
247414	0	0	1	0	0	1	1	0	0	1
nd	3	6	9	8	4	6	4	5	5	4
nw	3	6	9	8	4	6	4	5	5	4
nw/nd	1	1	1	1	1	1	1	1	1	1

Table 2.6: Matrix for greedy algorithm

At the beginning, we have calculated node degree(nd), node weight(nw) and new node value (nw/nd). Column 247112 is selected, and rows covered by this selected column will be removed and the matrix is updated.



	247173	247280	247281	247282	247283	247284	247401	247402	247414
247280	1	1	1	1	1	1	1	1	1
247282	0	1	1	1	1	0	0	0	0
247283	0	1	1	1	1	1	0	0	1
247284	0	1	0	0	1	1	0	0	1
247401	1	1	1	0	0	0	1	1	0
247402	1	1	1	0	0	0	1	1	0
247414	0	1	0	0	1	1	0	0	1
nd	6	9	8	4	6	4	5	5	4
nw	6	9	8	4	6	4	5	5	4
nw/nd	1	1	1	1	1	1	1	1	1

Table 2.7: Matrix after removing 1st column

Again, the process will get repeated and the solution for the greedy algorithm will be as follow

Node	***Nd***	***Nw***	***Nw/nd***	***Cost***	***Uniquerows***
247112	3	3	1	3	3
247173	6	6	1	6	6
247280	9	9	1	10	10

Table 2.8: Results for Greedy Algorithm

CHAPTER 3

REVIEWS

This chapter discusses the set covering algorithms, especially its greedy approach, weighted greedy approach and previous work related to the problem.

3.1 Set covering Algorithm

To know about what exactly set-covering algorithm is, we need to know about what set covering problem means and its desired solution.

According [13], the main idea behind the set cover problem is to select the minimum number of sets, so that those set covers all the elements of the original set with a minimum cost.

Input

Universal set U = {*u1, u2, u3… uN*}

Subset S1, S2, S3…Sk ⊆ U

Cost c1, c2, c3… cN

Goal

Find a set $I \subseteq \{1, 2, 3 \ldots n\}$ minimizing cost $\sum_{i \in I} c_i$ such that $\bigcup_{i \in I} Si = U$

Another definition of set covering problem is the followings:

Let A=(a_{ij}) be a 0-1 m x n matrix, and c= (c_j) be an n-dimensional integer vector. The value c_j (j ∈ N) represents the cost of column j, assuming that $c_j > 0$ for j ∈ N. j ∈ N covers a row i ∈ M if $a_{ij=}1$. SCP calls for a minimum-cost subset S of columns, such that each row i ∈ M is covered by at least one column j ∈ S. A natural mathematical model for SCP is

$$\min \sum_{j \in N} c_j x_j \qquad (1)$$

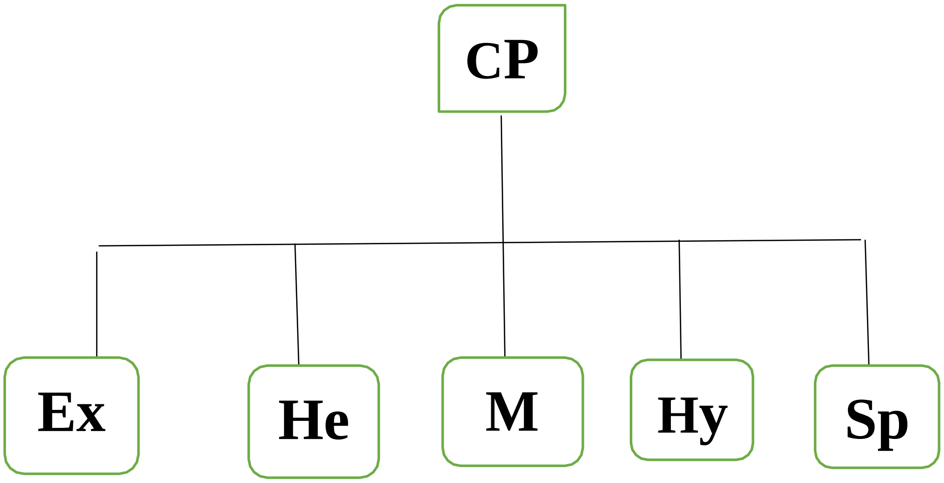
$$\text{Subject to} \sum_{j \in N} a_{ij} x_j \geq 1, \text{for all } i \in M, \qquad (2)$$

$$x_j \in \{0,1\}, j \in N \qquad (3)$$

Here Equation (2) makes sure that each row is covered by at least one column and Equation (3) is the integrality constraint [15].

3.2 Related Work

3.2.1 Classification of techniques for covering problem



The exact procedure (Ex) is intended to achieve the optimal solution for the SCP. This category includes dynamic programming, branch and bound techniques. Heuristic (He) aims to achieve near-optimal solution also known as single pass heuristic, which means that the procedure will end very quickly on the basis of set guidelines if the objective function is not improved. Meta-Heuristic (M) contains tabu search, simulated annealing, Genetic Algorithm (GA), etc. all aiming to achieve global optimum for the given problem. Hybrid heuristic (Hy) is formed by combining two or more of the heuristic. Special techniques (Sp) consist of the techniques for the covering problem which are not coming under any of these categories.

3.2.1.1 Exact Procedures

Toregas et al. (1971) a linear programming model has been developed to solve the traditional SCP with equal cost in the objective. Patel (1979) used a dynamic programming approach to locate rural social service centers for the Dharmpur village in India.

Chan and Yano (1992) a branch and bound algorithm have been developed with multiplier adjustment for the traditional SCP.

Williams (2005) addressed the maximal covering sub-tree problem that applies to the design of the transport network and extensive facility location. Two objectives are involved in finding an optimal sub-tree, viz. minimizing the total arc cost or distance of the sub-tree and maximizing the sub-trees coverage of population or demand at nodes. The author also presented four new biobjective zero-one programming models, which have ‘integer-friendly’ solution properties and are relatively small in terms of the number of decision variables and constraints.

Murry (2005) developed a model using the spatial structure to address complementary partial area services. The service coverage has been a fundamental aspect of geographic research. In particular, facility placement and associated coverage are central concerns in emergency services, transit route design, cartographic simplification, natural resource management and weather monitoring among others. The author also discussed the use of SCP, in geographic analysis. Problematic aspects of set coverage modelling across space are identified. The geographic information systems and emphasized spatial information have accentuated spatial representation issues that need to be taken into the account in the modelling of service coverage.

3.2.1.2 Heuristic

Since the covering problem is combinatorial in nature, heuristic development is inevitable. Kuehn and Hamburger (1963) developed a heuristic program to locate a warehouse as a covering problem.

Shannon and Ignizio (1972) developed a heuristic programming algorithm for warehouse location by taking into account the travel costs between the warehouse and the plants. This algorithm is based primarily on the add-and-drop technique. The author has assumed the upper limit for the number of warehouses to be located.

In order to determine the economical number of manufacturing cells and cell arrangements, Panneerselvam and Balasubramanian (1985) developed a set covering heuristic. The algorithm addresses the total facilities design problem including machine grouping, cell layout, cell loading and estimation of machine requirements and its impact on idle time and overtime of the machines.

Karmarkar et al. (1991) presented an interior point algorithm to solve computationally difficult SCPs. The interior point approach to the feasibility problem of 0-1 integer programming is based on the minimization of a non-convex potential function. The procedure generates a sequence of strict interior points of a polytype defined by a set of inequalities so that each consecutive point reduces the value of the potential function. The algorithm has an in-built module for two routines schemes to show how to proceed when a non-global local minimum is encountered.

El-Darzi and Mitra (1995) reviewed graph theoretic relaxations of the SCP and set partitioning problem. The greedy algorithm has been developed by the authors with a matching relaxation and a graph covering relaxation.

Haddadi (1997) proposed from a classic idea a simple Lagrangian heuristic for the set cover problem. It produces an efficient solution only by the low-density set cover problems, which is its limitation.

Paschos (1997) did a literature review on approximation minimum set covering, the minimum vertex covering, the maximum set packing and the maximum independent set problem. The author also discussed their approximation performance and their complexities.

Four serial heuristics and four parallel heuristics were observed by Chakarvarty and Shekhawat (1992) for the minimum set cover problem. These algorithms perform a trade-off between the run time and solution quality. The parallel heuristics are derived from the serial heuristics, where it is shown that an increase in a number of processors does not degrade the solution quality.

Flores et al. (1999) studied the test set compaction problem, which in digital system testing is a fundamental problem. The author has also studied the use of set covering models to the compaction of test sets, which can be used with any heuristic test compaction procedure .Effective set covering algorithm has been used for this purpose.They found that using the set co vering algorithms, the size of the computed test sets can often be reduced.

Chuzhoy and Naor (2002) considered the classical vertex cover and set cover problems with the addition of hard capacity constraints. This means that the set can only cover a limited number of its elements and the number of copies available for each set is bounded. The author has developed a 3-approximation algorithm which is based on randomized rounding with algorithms for unweighted vertex covering problem with hard capacities. They have shown that the weighted version is at least as difficult as a set cover problem.

Tsuyoshi and Toshihiro (2002) developed a modified version of the greedy algorithm for a set cover problem with two weights. They showed two sets of weights, viz. subsets weights restricted to one and a constant weight ‘d’. The algorithm produces the same approximation bound.

3.2.1.3 Meta-Heuristic

A local search heuristic for large SCP is provided by Jacobs and Brusco (1995).This heuristic is based on the simulated annealing algorithm and uses an improvement routine that provides lowcost solutions within a reasonable amount of CPU time.

In its genetic algorithm approach, Huang et al. (1994) considered the SCP's new penalty function and mutation operator to deal with the constraints. While approaching the optima, the mutation operator approaches on either side of the feasible or infeasible borders. This actually reduces the search time to half. The author used the new penalty function for handling the constraints.

By modifying the basic genetic procedure, including a new fitness crossover operator, Beasley a nd Chu (1994) developed a genetic algorithm based heuristic for non - unicost SCP, a variable mutation rate and a heuristic feasibility operator tailored specifically for the set cover problem. The heuristic is suitable to generate an optimal solution for problems of small size, which is its limitation.

Lorena and Lopes (1997) developed and applied a genetic algorithm to computationally difficult SCPs. This genetic algorithm implementation reaches high-quality computational results for difficult SCPs, arising in computing the 1-width of incidence matrix of Steiner triple systems.

For the SCP proposed by Catalano and Malucelli (2001), the parallel randomized heuristic is an i terative and an embedded constructive heuristic in a randomized procedure.The first heuristic g roup is obtained by randomizing the choices made at each stage of the construction of the solutio n. The second heuristic group is obtained by introducing random perturbation of the cost of the problem instance. The author also discussed a different parallel implementation of the heuristic.

Yoichi et al. (2001) examined the covering problem applied to geographic feature analysis. They compared the geographical feature analysis of the SCP search area based on the genetic cro ssover operator with the standard genetic analysis using Genetic Local Search. The author concluded that the geographical feature analysis based on the crossover is the most powerful.

Ermis et al. (2002) developed a meta-heuristic vibrational genetic algorithm to solve the problem of covering location in continuous space where the demand centers are served independently from independent supply centers. It is more of a covering problem rather than the set covering location problem. The algorithm uses a vibrational mutation that periodically introduces a random amplitude wave into the population starting with the initial step of the genetic process.

3.2.1.4 Hybrid heuristic

Caprare et al. (1999) presented a Lagrangian-based heuristic for large-scale SCP and the same was refined to give the best solution. The algorithm uses sub-gradient optimization to reduce computing time, coupled with pricing techniques. It is a replacement algorithm for the GREEDY procedure.

A genetic algorithm for time-satisfaction based SCP is an integer programming for minimizing the total fixed cost, rather than set covering location problem. This is designed using a mixed genetic algorithm, strategies proposed by Yun-Feng et al. (2005).

Vasko et al. (2005) surveyed several hybrid algorithms for the assessment of computational performance, using genetic algorithms for the SCP. The Greedy Randomized Adaptive Search Procedure (GRASP) is designed by genetic algorithm and local neighbourhood search approach.

Laifenfeld et al. (2006) considered the problem of finding the minimum identifying code in a graph, a designated set of vertices in which the neighbourhood uniquely overlaps at any vertex on the graph in order to show that it is computationally equivalent to SCP. The author has presented an approximation algorithm, based on SCP greedy approach. The identification of the code problem is a special case of the test problem and thus the test- case approximation can be used to produce ’good-identifying’ codes. The entropy-based approximation applied to graphs gives different edge probabilities.

In order to find a minimal logical function in the design of logical circuits, Seda (2007) develope d an approach to genetic algorithm based heuristic to solve SCP. The author compared the result with the Quine-McCluskey method. The proposed algorithm is used in setting the parameter for minimizing the Boolean functions.

In order to solve the SCP using three techniques, Gouwanda and Ponnambalam (2008) develope d an evolutionary search technique viz mathematical model using LINGO, GA toolbox from MATLAB and ACO programmed in MATLAB. The author concluded that LINGO has performed well in solving SCP as an optimization tool, but the GA tool does not perform well despite its flexibility. The combination of all the three techniques for solving SCP takes more time to solve the problem.

3.2.1.5 Special Techniques

Solar et al. (2002) showed a Parallel Genetic Algorithm (PGA) model to solve the SCP. Experimental results obtained with a binary representation of the SCP show that in terms of the number of generations needed to achieve solutions of acceptable quality, PGA only evaluates a pth part of the global population, instead of the process followed in the sequential genetic algorithm.

3.2.2. Deep Web Crawling Using a New Set Covering Algorithm

Wang et.al. [2] introduced a new algorithm for deep web crawling. To understand this concept, we first need to know the hidden web, deep web crawling and query selection.

Deep Web

Deep Web [26], also known as the invisible web is generated from the data source such as database or file system, and cannot be accessed by the people through URLs. Most of the people use various search engines like google for searching information on the web. The deep web is much larger than the surface web [25]. According to [27], the hidden web comprises nearly 550 billion documents which are 2,000 times greater than surface web.

Deep Web Crawler

Fetching the information from the hidden web is termed as deep web crawling. It helps in web indexing. Web crawler [28], [29], [30] crawls one page at a time through websites unless all pages are indexed. This helps in collecting data about links and website related to them, meta tag information, the web page content and much other information related to it. It also keeps track of URLs being already downloaded to avoid same page downloading again. They help in sending the queries against the index and provides the webpage matching the query.

Query Selection

Query selection is the process of selecting queries in such a way that they cover the maximum number of documents. There are many existing works facing this particular problem like [31], [32], [33], [34].

A solution related to this problem can be a random selection of some words from a dictionary. But, this solution may not be efficient because a large number of queries may not match with some pages, or may cause too many overlapping returns. Recently, several algorithms [31], [32], [33] have been developed for selecting queries from the downloaded documents, instead of selecting queries from a dictionary, that are retrieved by previously submitted queries to the deep web data source.

As we got a brief idea about the hidden web, deep web crawler, and query selection, consider the paper [2]. In this paper, the author presented a new algorithm named weighted greedy algorithm for the set covering along with greedy algorithm. The major task was to select the set of queries with low cost. As it was impossible to select queries directly from a large database, one solution was to select the queries from the sample of the database. It was shown that queries selected from the sample database performed well on the original database and also on a sample [31].

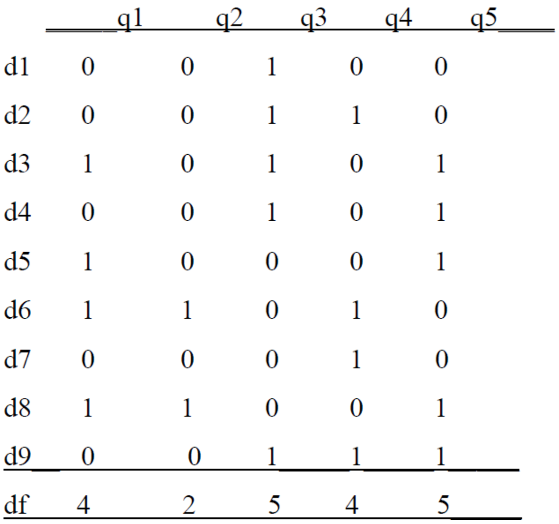
They created a four-step framework for deep web crawling:

i. Randomly select documents from original database to build a sample.

ii. Create a query pool i.e. a set of queries based on sampleDB.By using sampleDB and query pool, select a proper set of queries. iv. Map those selected queries to the original database.

Weighted Greedy Algorithm

The following step by step process leads to covering different rows at each step. An interesting question is: Does it make a difference in covering a row earlier or later? In a greedy strategy, all newly covered rows are considered having a unit cost, and there is no difference in whether they are being covered earlier or later.



	q1	q2	q3	q4	q5
d1	0	0	1	0	0
d2	0	0	1	1	0
d3	1	0	1	0	1
d4	0	0	1	0	1
d5	1	0	0	0	1
d6	1	1	0	1	0
d7	0	0	0	1	0
d8	1	1	0	0	1
d9	0	0	1	1	1
df	4	2	5	4	5

Table 3.1: Matrix A

In the above table 1, as we can see row d1 and d7 are getting only covered by q3 and q4 respectively. Due to this, they should get covered first as soon as possible. To understand properly, let’s say q4 is assigned as initially selected column and then the unique solution is {q4, q5, q3} with the cost 14. But the optimal solution can be {q4, q3, q1} with the cost 13. We can reach this optimal solution if q3 or q4 is set as an initial column. The greedy method failed to find an optimal solution, in this case, because it does not consider covering d1 and d7 by using q3 and q4 as early as possible [34].

Now we came to know there is a variation between covering elements earlier or later. The second question is how to measure such variation for covering a row earlier or later. Let’s say for each row i, it was shown in [12] that if the number of columns that row i can cover is larger, it is better to cover it later.

The two main reasons behind it:

- When row *i* is covered at a high coverage (in later steps) and most of the rows are already covered, more columns covering row *i* means that there could be more possibilities to select a small-cost column which covers few new rows (of course, at high coverage, no column can take many new rows) [12].
- When row *i* is covered at a low coverage (in earlier steps) and most of the rows are not covered yet, more columns covering row *i* means that there are more possibilities to cause overlapped coverage, i.e., row *i* will be covered many times.

Now, we consider *qw(j)* a good measurement than *new(j)* because it gives information regarding (1) how many documents can be gained by selecting query *qj* and (2) how quickly the document should be considered to be covered. Thus, the new/cost has been replaced by *qw/cost* for query selection. The related algorithm is called Weighted Greedy Algorithm.

Based on the above definition for document and query weight, [34] presented the Weighted Greedy Algorithm shown as Algorithm 2.

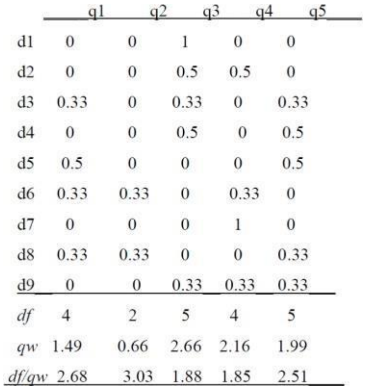
Algorithm 2: Weighted greedy algorithm.

Input: SampleDB, Query Pool QP, m x n Matrix A, where m= |SampleDB| and n= |QP| **Output**: a set of queries Q

Process:

1. Q = null
2. Let B =(b_{ij}) be a m x n matrix and $b_{ij} = a_{ij}$ x $dw_{d_i}^{QP}$;
3. Based on the matrix B, we calculate the query weight for each term and select q_j that minimizes $\frac{df_j}{qw_{q_j}^{QP}}$ into Q ;
4. Check if the queries in *Q* can cover all documents in Sample DB. If yes, the process ends;
5. Update matrix *B* by removing the selected query and the documents that are covered by the query. Go to Step 3.

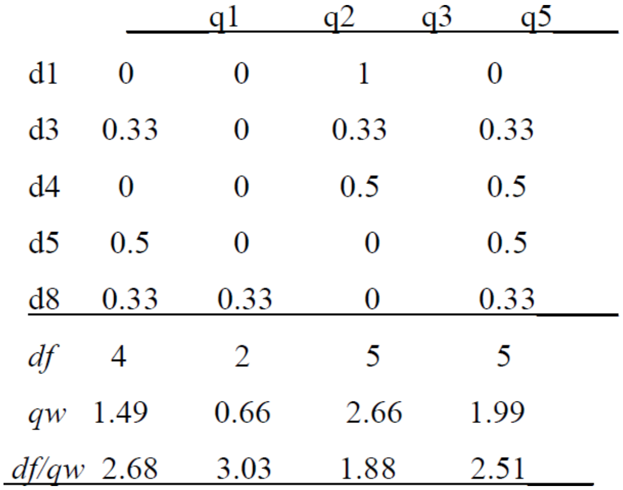
Example Based on the matrix shown in Table 1, the initial weights of the documents and the queries are given in Table 2. In the first step of Algorithm 2, row q4 holds the maximum value of qw = cost (1.85). It is selected as the first query, hence the corresponding q4 is set to 1. For convenience to the explanation, the column q4 and the covered rows, i.e., d2, d6, d7 and d9 are removed from the matrix A, and the resulting corresponding weighted matrix is shown in Table 3.



	q1	q2	q3	q4	q5
d1	0	0	1	0	0
d2	0	0	0.5	0.5	0
d3	0.33	0	0.33	0	0.33
d4	0	0	0.5	0	0.5
d5	0.5	0	0	0	0.5
d6	0.33	0.33	0	0.33	0
d7	0	0	0	1	0
d8	0.33	0.33	0	0	0.33
d9	0	0	0.33	0.33	0.33
df	4	2	5	4	5
qw	1.49	0.66	2.66	2.16	1.99
df/qw	2.68	3.03	1.88	1.85	2.51

Table 3.2: Initial Weight table of Matrix A

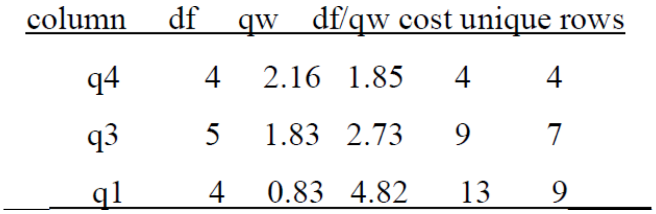
In the second and third steps, q3 and q1 are selected respectively and the solution of the weighted greedy is X = (1; 0; 1; 1; 0), and its cost is 13(4+5+4).



	q1	q2	q3	q5
d1	0	0	1	0
d3	0.33	0	0.33	0.33
d4	0	0	0.5	0.5
d5	0.5	0	0	0.5
d8	0.33	0.33	0	0.33
df	4	2	5	5
qw	1.49	0.66	2.66	1.99
df/qw	2.68	3.03	1.88	2.51

Table 3.3: Second step weight table of Example

As it is shown in table 3, only 4 queries and five documents are left now, the process will repeat until all the queries are covered. Table 4 shows the final result.



column	df	qw	df/qw	cost	unique rows
q4	4	2.16	1.85	4	4
q3	5	1.83	2.73	9	7
q1	4	0.83	4.82	13	9

Table 3.4: Result of example for a weighted greedy method

Here, queries q4, q3 and q1 are selected in each step respectively. In the table above, the cost is the total number of documents the query can fetch. The unique rows denote the number of total unique documents fetched by the query.

2.2.3. TS-IDS Algorithm for Query Selection in the Deep Web Crawling

In paper [8], the authors introduced a new algorithm for deep web crawling. The paper states that the importance of documents depend on the size of the terms and on the number of terms it contains. The size of a term is the number of documents the term can cover or document frequency. If there are less terms in a document, it can be covered with less redundancy, so it is of less importance in query selection. If a document contains large terms, the covering cost will be more, due to large terms.

Document of this type is more important, and the importance is proportional to the term size within the document.

Based on the above understanding, TS-IDS algorithm was proposed for selecting the queries. It out performs both traditional greedy and IDS algorithm.

3.2.3.1 The TS-IDS algorithm The

Query Selection Problem

We are a given set of documents D = $\{D_1, D_2, \ldots, D_m\}$ and a set of terms T = $\{T_1, T_2, \ldots.., T_n\}$. Every document contains a set of terms. Each term covers a set of documents. Undirected bipartite graph G(D, T, E) is formed by documents and terms where each node is in D or T, E is a set of edges between T and D (E ⊆ T × D). There is an edge between a document and a term iff the term is present in the document. The document-term matrix A = (a_{ij}) is used to represent the graph where

$$a_{ij} = \begin{Bmatrix} 1, & if\ Tj\ occurs\ in\ Di; \\ 0, & otherwise. \end{Bmatrix}$$

Let d_i^D and d_j^T represent the degrees of the document D_i and term T_j (the size of document and term) resp, where $d_i^D = \sum_{k=1}^{n} a_{ik}$, $d_j^T = \sum_{k=1}^{m} a_{kj}$.

For each document D_i, document weight is denoted by w_i. The weight of document D_i, is proportional to the minimal term size of the terms connected to D_i, and inversely proportional to its document size, i.e.,

$$w_i = \frac{1}{d_i^D} \min_{T_j \in D_i} d_j^T$$

From the above definition, the TS-IDS can be expressed in Fig. Here, m' is the number of documents in the new matrix after removing the covered documents. The new document weight of a term T_j, denoted by μ_j, is the sum of all document weight containing of term T_j i.e., μ_j =

$\sum_{i=1}^{m} a_{ij}\ w_i$. Comparing this with the μ_j used in the greedy algorithm where $\mu_j = \sum_{i=1}^{m\prime} a_{ij}$, the only difference between TS-IDS and greedy algorithm is the a weight of the documents i.e., w_i.

Algorithm 3: TS-IDS algorithm.

Input: SampleDB, Query Pool QP, m x n Matrix A, where m= |SampleDB| and n= |QP| **Output**: a set of queries Q

```
$x_j = 0, for\ 1 \le j\ \le n;$
$\mu_j = \sum_{i=1}^{m} a_{ij} w_{ij}, for\ 1\ \le j\ \le n;$
while not all docs covered do
    Find j that maximizes $\mu_j / d_j^T$;
    $x_j$ = 1;
    Remove column j;
    Remove all rows that contain $T_j$;
    $\mu_j = \sum_{i=1}^{m\prime} a_{ij}\ w_i$ in the new matrix
End
```

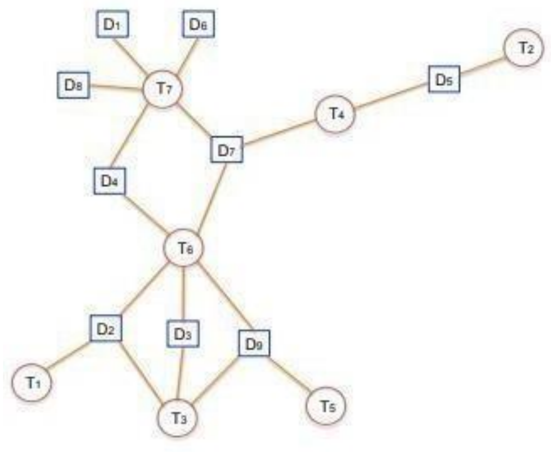


Fig 3.5: - A deep web data source is modelled as a bipartite graph [8]

In the above example, as we can observe, document 7 and 6 has degree 3 and 1 respectively, depicting that document 7 has more chances to get captured then document 6. Accordingly, document D_6 is more important that D_7. According to the definition of TS-IDS, D_6 will be captured first, and D_7 will be captured more than once. Considering a term, say T_4 in the solution set ,we ca n observe in the figure that D_7 contributes only one third portion of new document, as D_7 can be covered again by other terms. Hence, the importance of document D_i is inversely proportional to size of the document d_i^D (IDS).

Therefore, we are going to apply this algorithm to operational research and test its performance for comparison between various greedy approach's to see if TS-IDS gives better results in other fields also.

3.2.4. Covering problem in Facility Location

There are many existing covering problems, where customers receive services depending on the distance between the customer and the service provider. In covering problem, services provided

The first covering problem introduced in [41] was about the location of the center of a network, where the center of a network is the point of the network from which the distance to the furthest point is a minimum. The purpose of the model was to cover the node depending on the number of police required on the network highway, which was an application of vertex covering problem. The first mathematical model was introduced [41] used for the allocation of emergency service facilities. This is an application of set covering problem with equal costs. The sets consist of the facility points reachable within a specified time or distance. One constraint is presented for each demand point requiring to be covered and linear programming is used to solve the covering problem.

There are many books that have dealt with covering problem in details and among all, [38] has done a significant contribution. They divided the covering problem into two major categories: network location problem and cyclic network problem.

This paper [44] gave a solution for locating emergency medical vehicle service at all the sites to be covered, so that users at each point in the graph is able to find a service. They followed an approach where it first covered all those locations which are less accessible, and covered later on those locations which could be covered by many services. This work ensures the coverage of all the locations in the network while minimizing the number of facilities.

Another paper [41], showed how to decide 'good' locations for facilities on a network. It has gained a lot of attention in the last few years. This paper is about the facility-location problem with the special constraint on the maximum time or distance a user can spend to reach the closest service. The main problem stated in this paper is to minimize the total number of service facilities required to meet the response time or distance standards for each of the users. This standard is identical for all facility locations in terms of cost. The solution is the number and location of the facilities that can provide the desired service.

The authors presented an example of locating fire stations. Once a response time is given, then for every point in demand, there must be a fire station located within s time units. It is assumed that each facility has response capability at all times. The desired solution is to locate the minimum number of fire stations that satisfy the response-time requirement.

CHAPTER 4

EXPERIMENT

In this chapter, we present our experimental results obtained from various greedy approaches. The purpose is to test various greedy approaches to find which one is better than the others in our setup.

4.1 Dataset

The experimentation is performed on several regional maps obtained from the dataset. The dataset is of 1,965,206 nodes. We use approximately 5,000 nodes for our experimentation. The input is of two different types: one has coverage radius 1 and other has coverage radius greater than 1. Various greedy approaches have been implemented and run on 50 regional map with coverage radius 1 and the other 50 maps with coverage radius greater than 1. We will check the improvement of various greedy approaches in different situations like different coverage radius and different cost definitions.

The approaches are performed on approximately 5000 x 5000 nodes matrix. We have executed the corresponding code on the different coefficient of variance ranges. Here, the CV represents the spreading of node coverage degree.

The below graph presents the out degrees of the nodes in two situations: one with coverage radius 1, and other with coverage radius larger than one. The table below presents the two different sets of sample data. The results are for 50 sample maps, each with approximately 5000 rows and 5000 columns.

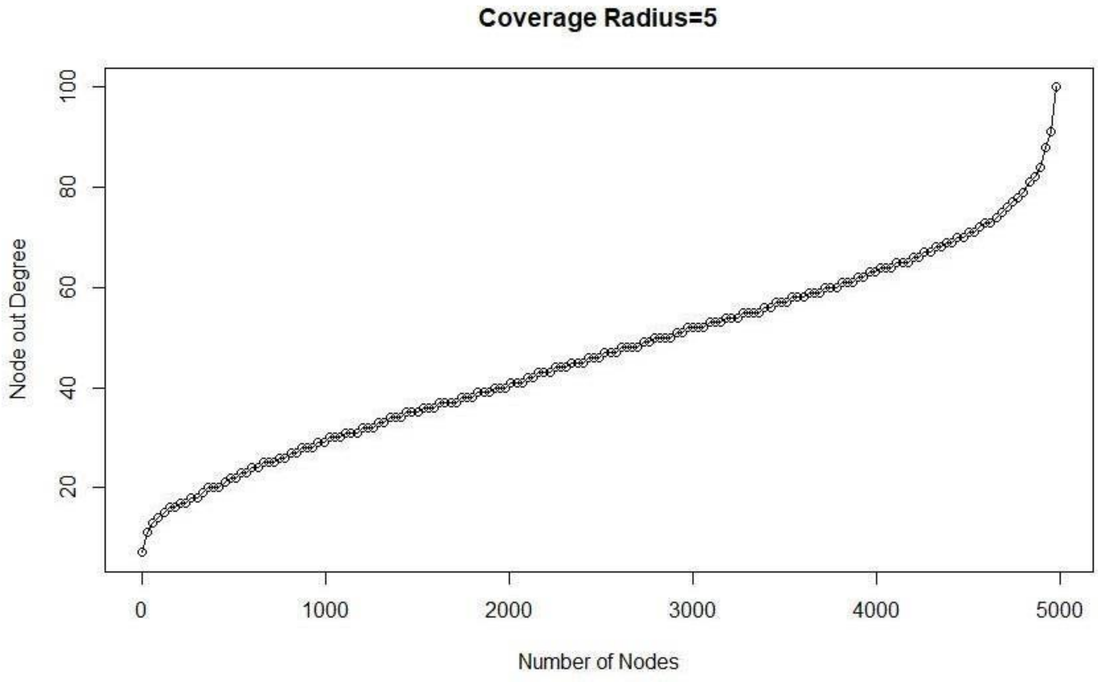


Fig 4.1: Distribution of coverage degree on radius 5

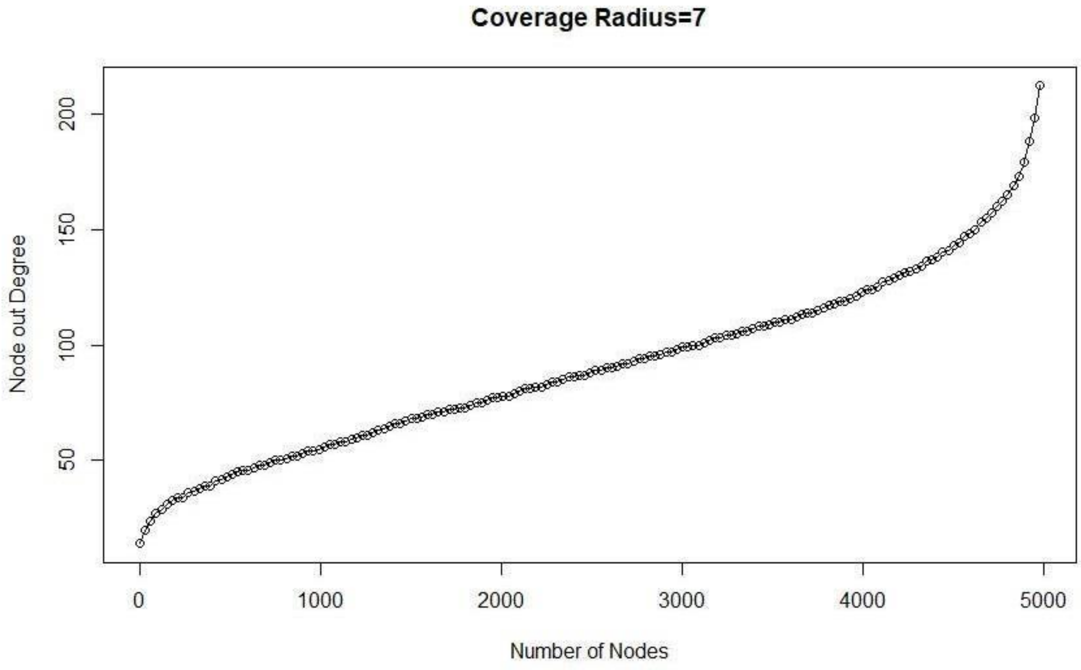


Fig 4.2: Distribution of coverage degree on radius 7

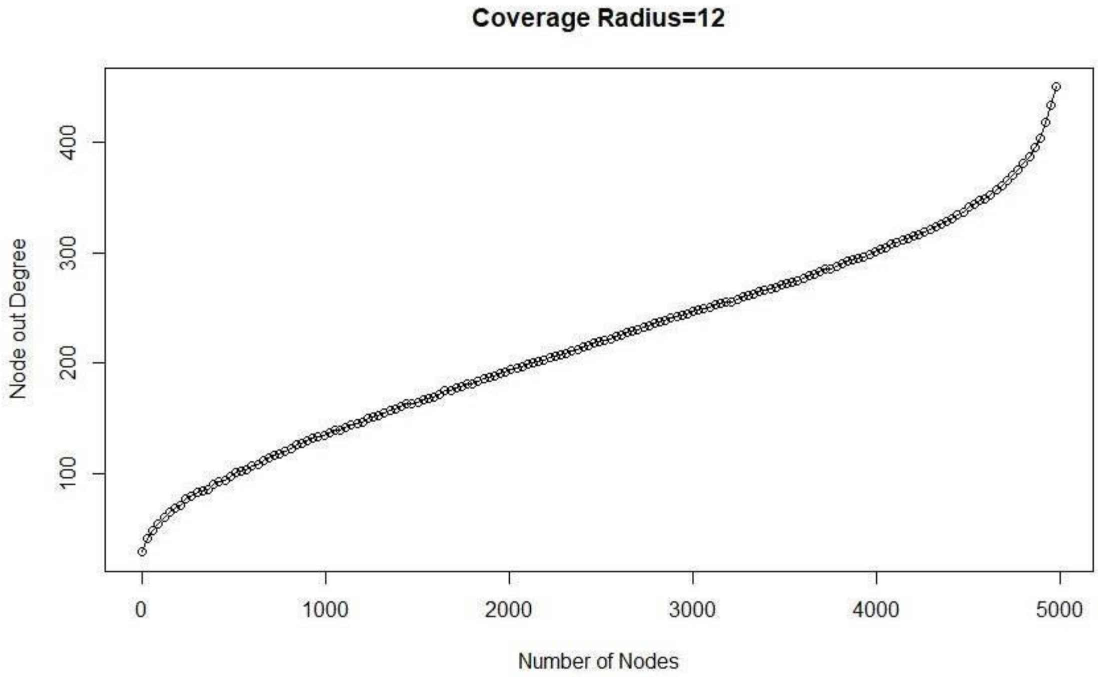


Fig 4.3: Distribution of coverage degree on radius 12

The above graph is the distribution of node degrees i.e., out degrees of the nodes under different coverage radius 5, 7, 12. The table shows the maximum and minimum out degree obtained from the given coverage radius. The results are taken from 5000 by 5000 rows and columns. Table 4.4 shows the maximum, minimum node degree.

In above figures 4.1, 4.2 and 4.3 we present the distribution of node degree on different coverage radius, where the x-axis represents the node id in the sample matrix and the y-axis represents the out-degree of each node.

In table 4.4, the maximum out-degree column and minimum out-degree column shows the maximum and the minimum number of out-degree for each coverage radius.

Coverage Radius	Maximum Out Degree	Minimum Out Degree
5	119	7
7	246	14
12	486	29

Table 4.4: Distribution of Node

4.2 Cost Definition

We will use different cost definitions to see how the performance of the algorithm will be affected. There are two types of cost definitions: -

Infrastructure Cost:

With this type, the cost of selecting a node is one. The problem is similar to the traditional set cover problem, in which the goal is to minimize the number of nodes selected. This is also known as a set covering with unit cost. Examples include the cost for building a school, food services or fuel station.

Location Cost:

With this type, the cost of the node is the out-degree of the node. For example, a mobile phone operator wants to provide service to a currently uncovered geographical region. Six locations are being considered for installation of towers for this purpose. Hence the out degree for each location will be ten if ten villages are getting covered by each location and cost will be ten.

4.3 Results

The data is separated into two parts, each with two subparts. The first part is with coverage radius 1 and the other with a radius greater than 1. Then we test different approaches of greedy on these two categories using two different frameworks of cost definitions.

We used 50 sample maps with coverage radius 1 and 50 maps with a radius greater than 1. Then different greedy approaches are tested ten times on each sample map using cost definitions and the average value is taken from the solution nodes of each approach and then it is reported. We run all the algorithms, ten times in order to check how many numbers of solution nodes are getting selected at each execution and also to check their performance on each cost definition.

The improvement is defined by the cost of greedy (C_g) subtract the cost of various greedy approach’s and divided by the cost obtained from the greedy.

$$\text{IMP} = \frac{C_g - C}{}$$

Fig 4.5: Formula for calculating Improvement

In our experiment, the new TS-IDS algorithm gave better results on our dataset as the following tables and figures show:

Cost calculated in all different scenarios by the TS-IDS algorithm is always less than that from the greedy algorithm.

For a better analysis of the result, we took four random rows from the 50 improvement results.

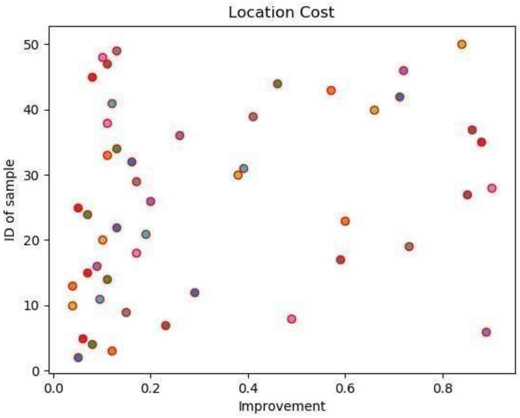


Fig 4.6: Greedy Algorithm vs TS-IDS algorithm, in Location Cost

Coverage Radius	Greedy Algorithm	TS-IDS Algorithm	Improvement %
3	207	182	**12**
7	198	152	**23**
10	535	400	**25**
15	576	553	**39**

Table 4.7: Greedy Algorithm vs TS-IDS algorithm, in Location Cost

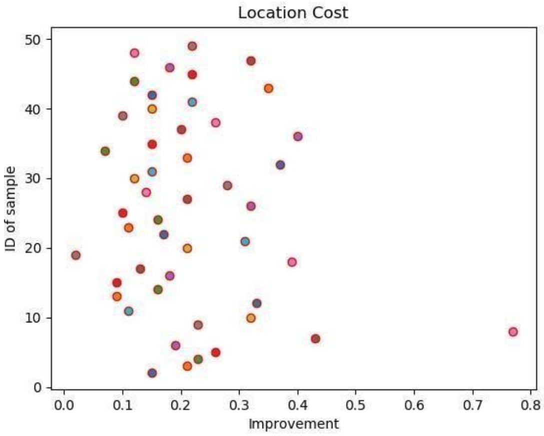


Fig 4.8: Greedy Algorithm vs other Approach, in Location Cost

Coverage Radius	Greedy Algorithm	Other Approach	Improvement %
3	207	163	**21**
7	233	171	**26**
10	470	336	**28**
15	198	112	**43**

Table 4.9: Greedy Algorithm vs other Approach, in Location Cost

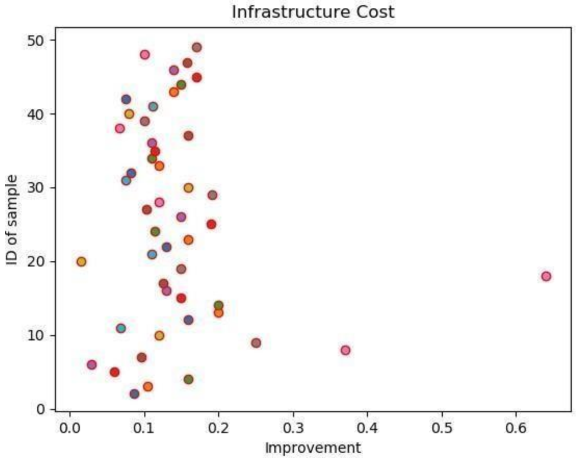


Fig 4.10: Greedy Algorithm vs TS-IDS algorithm, in Infrastructure Cost

Coverage Radius	Greedy Algorithm	TS-IDS Algorithm	Improvement %
3	358	333	**6**
4	453	415	**8**
6	384	334	**12**
8	321	276	**14**

Table 4.11: Greedy Algorithm vs TS-IDS algorithm, in Infrastructure Cost

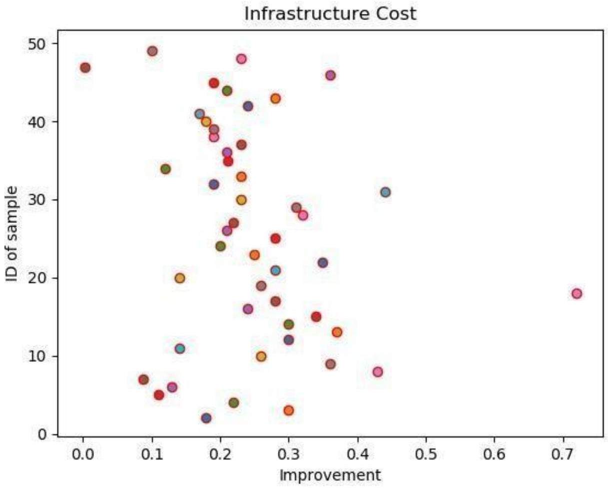


Fig 4.12: Greedy Algorithm vs other approach, in Infrastructure Cost

Coverage Radius	Greedy Algorithm	Other Approach	Improvement %
3	358	317	**11**
4	341	416	**17**
6	218	172	**21**
8	229	175	**23**

Table 4.13: Greedy Algorithm vs other approach, in Infrastructure Cost

We run three algorithms, the Greedy, the TS-IDS algorithm and the other approach. Fig 4.6 and fig 4.8 show the comparison of these algorithms in terms location cost fig 4.10, fig 4.12 is infrastructure cost. The TS-IDS algorithm outperforms the greedy algorithm in each execution by 31% in location cost. The other approach outperforms the greedy algorithm by 21% in location cost. Also in infrastructure cost, the TS-IDS outperforms the greedy algorithm by 14% and the other approach outperforms the greedy by 24%. Hence, TS-IDS algorithm and the other approach achieves better improvement than the greedy algorithm in every execution.

The result is the comparison of the greedy approaches that are executed ten times for the same dataset.

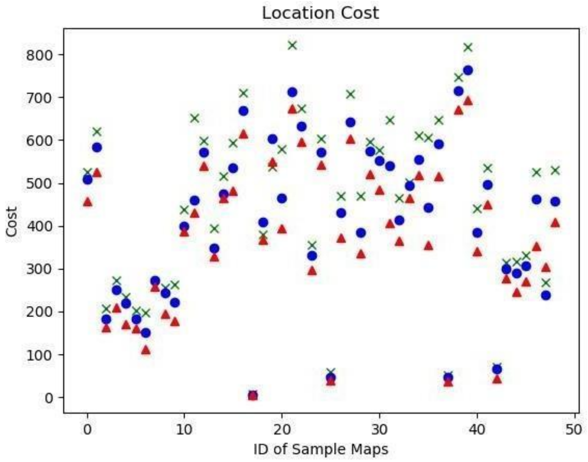


Fig 4.14: Location Cost

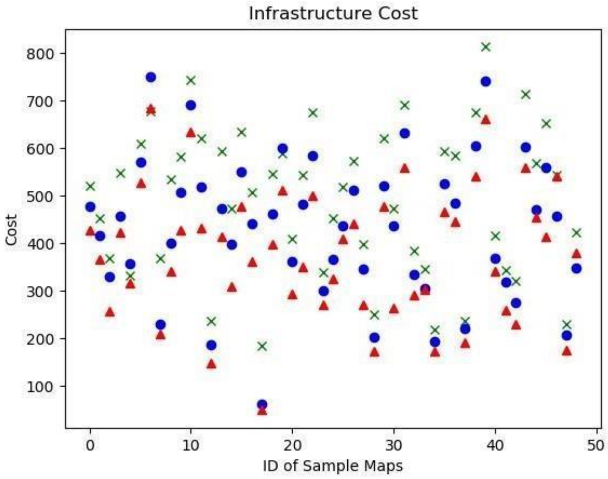


Fig 4.15: Infrastructure Cost

From fig 4.14 and 4.15, we can see that the average cost of the greedy algorithm is more than the other two approaches with different cost definitions. In the figure, the greedy algorithm forms larger number of solution sets than the others. The cost can be associated with location cost, cost of building facilities etc.

4.4 Data Distribution

The major idea in the two weighted greedy algorithms is that the rows should never be evaluated the same. There must be uniqueness in covering the different rows. If all rows are getting covered by the matching number of the columns, i.e., if the out-degree of each node is the same then all algorithms give the same result. That means, if the weight of all the nodes is the same, then TSIDS algorithm will be the same as of greedy algorithm. We can only predict the cost saving when there are different node coverage degrees. That is, the different node should have different coverage degrees. More the dispersion of node coverage degree, the larger improvement we get.

We calculate the improvement of the different greedy approaches to the CV. Once improvement has been calculated, the data is plotted on the graph that represents the improvement matched with CV.

The below figures show the relationship between the improvement of the various greedy approaches and the Coefficient of Variance for all sample maps.

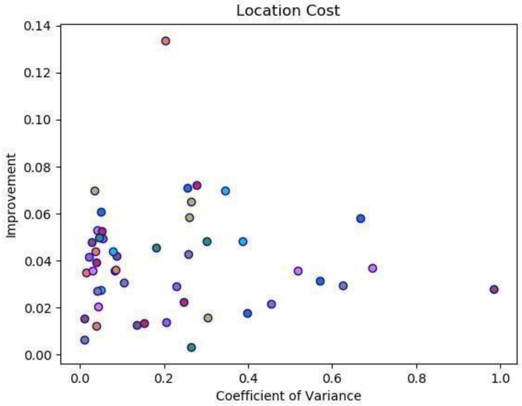


Fig 4.16: Relation between Improvement and CV for Location Cost

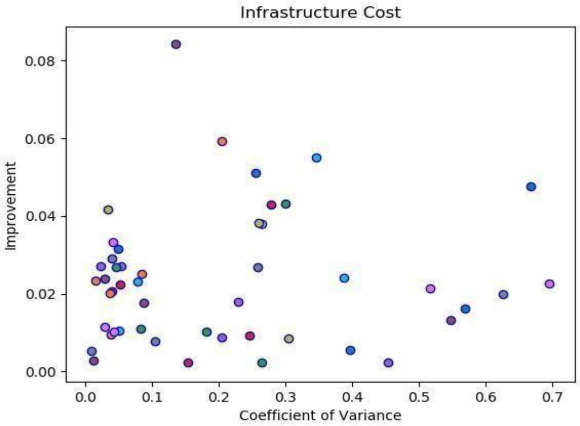


Fig 4.17: Relation between Improvement and CV for Infrastructure Cost

In the above figure, the x-axis represents the coefficient of variance points and the y-axis represents the improvement of TS-IDS algorithm over the greedy algorithm

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1. Summary of research and table

In this thesis, we tested different greedy approaches in the resource allocation field. The experiment is carried out using different cost definitions. The result shows that, same as in web crawling, we get better results than greedy algorithm. The results obtained from the two cost definitions show that this algorithm can be used when considering different cost definition.

TS-IDS algorithm was first designed for deep web crawling, but the results from our work convey that it can be used in other fields also. In our work, results were obtained using real data with different CV ranges.

The newly proposed algorithm outperformed both the greedy algorithm and the TS-IDS algorithm in terms of the number of solution nodes. The application of the proposed algorithm with its costeffectiveness can be useful in the field of resource allocation.

5.2. Future work

Due to a limited number of resources, allocating resources is a big problem. As we can observe, TS-IDS gave better results in our setup, but there are things that need to be considered in future improvement.

One aspect could be to test it in different fields of computer science to check if there are any other applications of the algorithm in other fields also. As there are various other applications of the set covering problem, it can also be tested in all those applications to check if it can work in other fields or not. We can also write some good algorithms, using the idea of this algorithm according to the requirement in the future.

BIBLIOGRAPHY

1. Cormode, G., Karloff, H., & Wirth, A. (2010, October). Set cover algorithms for very large datasets. In Proceedings of the 19th ACM international conference on Information and knowledge management (pp. 479-488). ACM.
2. Wang, Y., Lu, J., & Chen, J. (2009, August). Crawling deep web using a new set covering algorithm. In International Conference on Advanced Data Mining and Applications (pp. 326-337). Springer, Berlin, Heidelberg.
3. Mihail, M. (1999). Set cover with requirements and costs evolving over time. In *Randomization, approximation, and combinatorial optimization. algorithms and techniques* (pp. 63-72). Springer, Berlin, Heidelberg.
4. Golab, L., Karloff, H., Korn, F., Srivastava, D., & Yu, B. (2008). On generating nearoptimal tableaux for conditional functional dependencies. *Proceedings of the VLDB Endowment*, *1*(1), 376-390.
5. Lan, G., DePuy, G. W., & Whitehouse, G. E. (2007). An effective and simple heuristic for the set covering problem. European journal of operational research, 176(3), 1387-1403.
6. Cardei, M., Thai, M. T., Li, Y., & Wu, W. (2005, March). Energy-efficient target coverage in wireless sensor networks. In INFOCOM 2005. 24th annual joint conference of the ieee computer and communications societies. Proceedings ieee (Vol. 3, pp. 1976-1984). IEEE.
7. Farahani, R. Z., Asgari, N., Heidari, N., Hosseininia, M., & Goh, M. (2012). Covering problems in facility location: A review. *Computers & Industrial Engineering*, *62*(1), 368407.
8. Wang, Y., Lu, J., & Chen, J. (2014, September). Ts-ids algorithm for query selection in the deep web crawling. In *Asia-Pacific Web Conference* (pp. 189-200). Springer, Cham.

9. Lu, J., Wang, Y., Liang, J., Chen, J., & Liu, J. (2008, December). An approach to deep web crawling by sampling. In Web Intelligence and Intelligent Agent Technology, 2008. WIIAT'08. IEEE/WIC/ACM International Conference on (Vol. 1, pp. 718-724). IEEE.

10. Grossman, T., & Wool, A. (1997). Computational experience with approximation algorithms for the set covering problem. European Journal of Operational Research, 101(1), 81-92.

11. Beasley, J. E. (1987). An algorithm for set covering problem. *European Journal of Operational Research*, *31*(1), 85-93.

12. Farahani, R. Z., Asgari, N., Heidari, N., Hosseininia, M., & Goh, M. (2012). Covering problems in facility location: A review. Computers & Industrial Engineering, 62(1), 368407.

13. Korte, B., Vygen, J., Korte, B., & Vygen, J. (2012). Combinatorial optimization (Vol. 2). Heidelberg: Springer.

14. M. Mihail. Set cover with requirements and costs evolving over time. In Proceedings of the Second International Workshop on Approximation Algorithms for Combinatorial Optimization Problems: RANDOM-APPROX, pages 63–72, 1999.

15. B. Saha and L. Getoor. On Maximum Coverage in the streaming model & application to multi-topic blog-watch. In 2009 SIAM International Conference on Data Mining (SDM09), April 2009.

16. Beasley, J. E., & Jörnsten, K. (1992). Enhancing an algorithm for set covering problems. European Journal of Operational Research, 58(2), 293-300.

17. Bergman, M.K.: The deep web: Surfacing hidden value. The Journal of Electronic Publishing 7(1) (2001)

18. He, B., Patel, M., Zhang, Z.,&Chang, K.C.C. (2007). Accessing the deep web. Communications of the ACM, 50(5), 94-101.

19. Gupta, S., & Bhatia, K. K. (2014). A comparative study of hidden web crawlers. Ar Xiv preprint arXiv: 1407.5732.

20. Barbosa, L., Freire, J.: Siphoning hidden-web data through keyword-based interfaces. In: Proc. of SBBD (2004)

21. Chang, C.H., Kayed, M., Girgis, M.R., Shaalan, K.F.: A survey of web information extraction systems. IEEE Transactions on Knowledge and Data Engineering 18(10), 1411–1428 (2006)

22. Liddle, S.W., Embley, D.W., Scott, D.T., Yau, S.H.: Extracting data behind web forms. In: Olive, `A., Yoshikawa, M., Yu, E.S.K. (eds.) ER 2003. LNCS, vol. 2784, pp. 402–413. Springer, Heidelberg (2003)

23. J.Madhavan, D.Ko, L.Kot, V.Ganapathy, A.Rasmussen, and A.Halevy, “Google's deepweb crawl," in Proc. of VLDB, 2008, pp. 1241-1252.

24. S.Raghavan and H.G.Molina, “Crawling the hidden web,” in Proc. of the 27th international Conference on Very Large Data Bases (VLDB), 2001, pp. 129– 138.

25. A.Ntoulas, P.Zerfos, and J.Cho, “Downloading textual hidden web content through keyword queries,” in Proc. of the Joint Conference on Digital Libraries (JCDL), 2005, pp. 100–109.

26. Roth R (1969) Computer solutions to minimum cover problems. Operational Research 17:455–465

27. Toregas C, Swain R, ReVelle C, Bergman L (1971) The location of emergency service facilities. Operational Research 19:1363–1373

28. Church R, ReVelle C (1974) The maximal covering location problem. Pap Reg Sci 32:101–118

29. Mihelic, J., & Robic, B. (2004). Facility location and covering problems. In Proc. of the 7th International Multiconference Information Society (Vol. 500).

30. M. S. Daskin. Network and Discrete Location: Models Algorithms and Applications. Wiley, New York, 1995, Z. Drezner and H. W. Hamacher, editors. Facility Location: applications and theory. Springer-Verlah, Berlin, 2002

31. D. S. Hochbaum and D. B. Shmoys. A unified approach to approximation algorithms for bottlneck problems. J. of ACM, 33:533–550, 1986. PR.

32. Fallah, H., Sadigh, A. N., & Aslanzadeh, M. (2009). Covering problem. In Facility Location (pp. 145-176). Physica-Verlag HD.

33. Toregas, C., Swain, R., ReVelle, C., & Bergman, L. (1971). The location of emergency services facilities. Operations Research, 19, 1363–1373

34. Francis, R. L., & White, J. A. (1974). Facility layout and location an analytical approach (1st ed.). Englewood Cliffs, NJ, US: Prentice-Hall

35. Toregas, Constantine, and Charles ReVelle. "Optimal location under time or distance constraints." Papers in Regional Science 28.1 (1972): 133-144.

36. Peng, H., Qin, Y., & Yang, Y. (2016). Relationship between Set Covering Location and Maximal Covering Location Problems in Facility Location Application. In Proceedings of the 2015 International Conference on Electrical and Information Technologies for Rail Transportation (pp. 711-720). Springer, Berlin, Heidelberg.

37. Stanford Large Network Dataset Collection, snap.stanford.edu/data/, Accessed: 15 May 2017

38. Introduction to Python ,https://www.python.org/doc/essays/blurb/

39. Vazirani, V. V. (2001). Approximation Algorithms. Springer, Berlin, Germany.

40. Patel, N. R. (1979). Locating rural social service centers in India. *Management Science*, *25*(1), 22-30.

41. Chan, T. J., & Yano, C. A. (1992). A multiplier adjustment approach for the set partitioning problem. *Operations Research*, *40*(1-supplement-1), S40-S47.

42. Murray, A. T. (2005). Geography in coverage modeling: exploiting spatial structure to address complementary partial service of areas. *Annals of the Association of American Geographers*, *95*(4), 761-772.

43. Williams.J.C. (2003) 'Optimal direct and indirect covering trees', Annals of Operations Research, Vol. 123, No. 1–4, pp.265–284.

44. Kuehn, A.A. and Hamburger, M.J. (1963) 'A heuristic program for locating warehouse', Management Science, Vol. 9, pp.643–666.

45. Shannon, R.E. and Ignizio, J.P. (1972) 'A heuristic programming algorithm for warehouse location', AIIE Transactions, Vol. 2, No. 4, pp.334–339.

46. Panneerselvam, R. and Balasubramanian, K.N. (1985) 'Algorithmic grouping of operations sequences', Engineering Costs and Production Economics, Vol. 9, pp.125–134.

47. Karmarkar, N., Resende, M.G.C. and Ramakrishnan, K.G. (1991) 'An interior point algorithm to solve computationally difficult set covering problems', Mathematical Programming, Vol. 52, pp.597–618.

48. El-Darzi, E. and Mitra, G. (1995) 'Graph theoretic relaxations of set covering and set partitioning problems', European Journal of Operational Research, Vol. 87, No. 1, pp.109–121.

49. Haddadi, S. (1997) ‘Simple Lagrangian heuristic for the set covering problem’, European Journal of Operational Research, Vol. 97, No. 1, pp.200-204.

50. Paschos, V.T. (1997) ‘A survey of approximately optimal solutions to some covering and packing problems’, ACM Computing Surveys, Vol. 29, No. 2, pp.171–209.

51. Chakravarty, S. and Shekhawat, A. (1992) ‘Parallel and serial heuristics for the minimum set cover problem’, Journal of Supercomputing, Vol. 5, No. 4, pp.331–345.

52. Flores, P.F., Neto, H.C. and Marques-Silva, J.P. (1999) ‘On applying set covering models to test set compaction’, Proceedings of the Ninth Great Lakes Symposium on VLSI, 4–6 March, Ypsilanti, MI, pp.8–11.

53. Chuzhoy, J. and Naor, J.S. (2002) ‘Covering problems with hard capacities’, Proceedings of the 43rd Annual IEEE Symposium, pp.481–489.

54. Tsuyoshi, O. and Toshihiro, F. (2002) ‘A modified greedy algorithm for set cover problem with 2 weights’, IEIC Technical Report, Vol. 102, No. 90, pp.41–48.

55. Jacobs, L.W. and Brusco, M.J. (1995) ‘Note: A local-search heuristic for large set-covering problems’, Navel Research Logistics, Vol. 42, No. 7, pp.1129–1140.

56. Huang, W-C., Kao, C-Y., Horng, J-T. (1994) ‘A genetic algorithm approach for the set covering problems’, Proceedings of the first IEEE World Congress on Computational Intelligence, 27–29 June, Vol. 2, pp.569–574.

57. Beasley, J.E. and Chu, P.C. (1994) A Genetic Algorithm for the Set Cover Problem, Technical Report, The Management School, Imperial College, London.

58. Lorena, L.A.N. and Lopes, L.D.S. (1997) ‘Genetic algorithms applied to computationally difficult set covering problems’, Journal of the Operational Research Society, Vol. 48, No. 4, pp.440–445.

59. Catalano, M.S.F. and Malucelli, F. (2001) ‘Parallel randomized heuristics for the set covering problem’, Practical parallel computing, ISBN: 1-59033-127-3, pp.113-132.

60. Yoichi, I., Kengo, K. and Hiroyunki, N. (2001) ‘Efficient genetic operator in the setcovering problem’, Bulletin of the Okayama University of Science A. Natural Science, Vol. 37, pp.137–143, ISSN: 0285-7685.

61. Ermis, M., Ulengin, F. and Hacioglu, A. (2002) ‘Vibrational genetic algorithm (VGA) for solving continuous covering location problems’, Lecture Notes in Computer Science, Vol. 2457, pp.293–302, ISSN: 0302-9743.

62. Caprara, A., Fischetti, M. and Toth, P. (1999) ‘A heuristic method for the set covering problem’, Operations Research, Vol. 47, No. 5, pp.730–743.

63. Yun-Feng, M.A. Yang, C. and Zhang M. (2005) ‘Genetic algorithm for time-satisfactionbased set covering location problems’, Proceedings of the International Conference on Communications, Circuits and Systems, 27–30 May, Vol. 2, p.1041.

64. Vasko, F.J., Knolle, P.J. and Spiegel, D.S. (2005) ‘An empirical study of hybrid genetic algorithms for the set covering problem’, The Journal of the Operation Research Society, Vol. 56, No. 10, pp.1213–1223.

65. Laifenfeld, M., Tranchtenberg, A. and Berger-Wolf, T.Y. (2006) ‘Identifying codes and the set cover problem’, Proceedings of the 44th Annual Allerton Conference on Communication, Control and Computing, 27–29 September, Monticello, Illinois.

66. Seda, M. (2007) ‘Heuristic set-covering-based postprocessing for improving the Quine-McCluskey method’, International Journal of Computational Intelligence, Vol. 4, No. 2, pp.139–143, ISSN: 1304-2386.

67. Gouwanda, D. and Ponnambalam, S.G. (2008) ‘Evolutionary search techniques to solve set covering problems’, Proceedings of World Academy of Science, Engineering and Technology, Vol. 29, ISSN: 1307-6884.

68. Solar, M., Parada, V. and Urrutia, R. (2002) ‘A parallel genetic algorithm to solve the setcovering problem’, Computers and Operations Research, Vol. 29, No. 9, pp.1221–1235.

VITA AUCTORIS

NAME:	Shreeya Singhania
PLACE OF BIRTH:	Mumbai, Maharashtra, INDIA
YEAR OF BIRTH:	1994
EDUCATION:	Thakur Polytechnic, Diploma, Mumbai, INDIA,2013
	Thakur College of Eng. and Technology, B.E., 2016
	University of Windsor, M.Sc., Windsor, ON, 2019