APPROACHES TO MINIMUM SUM OF DIAMETER AND RADII CLUSTERING

Ph.D. Thesis

By

RAJKUMAR JAIN

DISCIPLINE OF COMPUTER SCIENCE & ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY INDORE

JULY 2018

APPROACHES TO MINIMUM SUM OF DIAMETER AND RADII CLUSTERING

A THESIS

Submitted in partial fulfillment of the requirements for the award of the degree
of
DOCTOR OF PHILOSOPHY

by
RAJKUMAR JAIN

DISCIPLINE OF COMPUTER SCIENCE & ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY INDORE

JULY 2018

INDIAN INSTITUTE OF TECHNOLOGY INDORE

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the thesis entitled **APPROACHES TO MINIMUM SUM OF DIAMETER AND RADII CLUSTERING** in the partial fulfillment of the requirements for the award of the degree of **DOCTOR OF PHILOSOPHY** and submitted in the **DISCIPLINE OF COMPUTER SCIENCE & ENGINEERING, Indian Institute of Technology Indore**, is an authentic record of my own work carried out during the time period from January 2011 to July 2018 under the supervision of Dr. Narendra S. Chaudhari, Professor, Discipline of Computer Science & Engineering, Indian Institute of Technology Indore.

The matter presented in this thesis has not been submitted by me for the award of any other degree of this or any other institute.

RKJain 30/12/20

signature of the student with date
(RAJKUMAR JAIN)

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

NChaudhari 30/12/2020

Signature of the Thesis Supervisor
(Dr. NARENDRA S. CHAUDHARI)

RAJKUMAR JAIN has successfully given his Ph.D. Oral Examination held on 30th December 2020

Signature of Chairperson
Date: Dec. 30, 2020

Prof. Rajeev Wankar
Digitally signed by Prof. Rajeev Wankar
Date: 2021.01.04 14:12:48 +05'30'

Signature of External Examiner
Date:

NChaudhari

Signature of Thesis Supervisor
Date: 30th Dec 2020

Signature of PSPC
Date: Dec. 30, 2020

Signature of PSPC
Date: Dec. 30, 2020

Signature of Convener
DPGC
Date: 30-12-2020

ACKNOWLDGEMENTS

Firstly, I would like to express my sincere gratitude to my supervisor Dr. Narendra S. Chaudhari, for the continuous support of my Ph.D. study and related research, for his guidance, encouragement, patience, motivation, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my Ph.D. study. This doctoral programme was possible only because of the unconditional support provided by Sir. I consider it as a excellent opportunity to do my doctoral programme under his guidance and to learn from his research expertise.

Besides my advisor, I would like to thank the rest of my thesis committee members Dr. Anand Parey and Dr. Ram Bilas Pachori for their suggestions, insightful comments, encouragement, and continuous motivation towards my research. My sincere thanks go to Dr. Surya Prakash, Dr. Aruna Tiwari, Dr. Abhishek Shrivastava, and Dr. Kapil Ahuja for discussion and motivation. Without their precious support, it would not be possible to conduct this research.

I would like to thank Shailendra Verma, Lalit Jain, Prahalad Singh Panwar, Dheeraj Verma from the Discipline of Computer Science and Engineering for their constant support and help. The thesis would not have come to a successful completion, without the help received from the staff of the Academic section. I would like to thank Tapesh Parihar and all other administrative staff members for their support at all times. I express my appreciation to my friends Jaya Thomas, Prakash Chandra Sharma, Neetesh Saxena, Varun Bajaj, Rajat, Rudresh, Rajendra and Sadaf all extended their support in a very special way. I want to thank Mrs. Swati Chaudhari for their support and constant encouragement.

Last, but not least, I would like to dedicate this thesis to my parents, my sisters, my wife and my brother, for their love, patience, and understanding; they allowed me to spend most of the time on this thesis.

Dated Rajkumar Jain

Dedicated to my parents

ABSTRACT

Cluster analysis is a key technique in the data analysis and is being applied in a variety of engineering and scientific disciplines such as biology, medicine, marketing, information retrieval and pattern recognition. Stereotyped clustering method propagates dissection effect. To thwart dissection effect, many real-time applications uses minimum sum of diameter clustering (MSDC) or minimum sum of radii clustering (MSRC). As, MSDC and MSRC problems are NP-complete so it is natural to seek approximation algorithms with the best provable approximation ratio. This thesis presents approaches for generating solutions for clustering problems. The thesis addresses clustering problems in three parts: (i) Comparative review of clustering algorithms (ii) SAT formulation-based approach, and (iii) Constraint based clustering approach

In the first part of this thesis: we investigated approach, distance criterion, optimization methods, geometrical properties, assumptions, issues, limitations, and time-bound to propose a comprehensive and comparative analysis of algorithmic complexity of various exact and approximation clustering algorithms for Euclidean, metric and geometric version of MSDC and MSRC problem. SAT formulation-based approach: In this approach, we investigate the technique for the reduction of the 3-cluster problem into 3-SAT and *k*-cluster problem into *k*-SAT.

In the constraint based clustering approach, we address three types of problems: 3-clustering problem for minimum sum of diameter, word clustering algorithm based on the *k*-clustering algorithm, and constraint based approximation algorithm for the min-cost *k*-cover problem. In the 3-clustering problem, we proposed a constraint algorithm for three clustering for the minimum sum of diameter problem. In the word clustering algorithm, a new constraint word clustering algorithm is proposed. The main challenge is to find out constraint for words having an asymmetric relationship between them. In

this context, we investigated properties of word cloud like symmetric, transitive, and implicative and also investigated various types of associations like strong, weak, and zero association between words. We proposed a constraint word algorithm based on the investigated properties and association. Furthermore, we bring the concept of constraints in the min-cost k-cover problem to improve the performance. Constraints help in reducing the number of distinct maximal discs. By incorporating the constraints into the min-cost k-cover algorithm, we proposed an approximation algorithm with the improved approximation ratio.

LIST OF PUBLICATIONS

International Journal

1. Jain R. and Chaudhari N.S. (2016), On Constraint Clustering to Minimize the Sum of Radii, International Journal of Engineering and Advanced Technology, 5(6), 236-239.

2. Jain R. and Chaudhari N.S. (2012), A New Bit Wise Technique For 3-Partitioning Algorithm, International Journal on Computer Application (IJCA) in Special Issue of International Journal of Computer Applications on Optimization and On-chip Communication, 1, 1-5.

3. Jain R. and Chaudhari N.S. (2011), Representation of K-Cluster Constraint as K-SAT in Social Networking, International Journal on Computer Application (IJCA) Special Issue of International Journal of Computer Applications on Evolution in Networks and Computer Communications, 1, 13-18.

International Conferences

1. Jain, R. and Chaudhari, N.S., (2015), Constraint Word Clustering Algorithm for Asymmetric Relationship, Proceeding of International Conference on Computer, Control and Communication Technologies, Thailand, pp. 109-115.

2. Jain, R. and Chaudhari, N.S., (2012), A New 3-Clustering Algorithm for Minimum Sum of Diameter Using Bit Representation, Proceeding of 7^{th} IEEE Conference International Conference Industrial Electronics and Applications (ICIEA), Singapore, pp. 2004-2009.

3. Jain, R. and Chaudhari, N.S., (2011), Identification and Generation of Constraints in Social Networks, Proceeding of IEEE International Level conference on Emerging Trends in Networks & Computer Communications, India, pp. 11-14.

4. Jain, R. and Chaudhari, N.S., (2011), Formulation of 3-Clustering as a 3-SAT Problem, Proceeding of Fifth Indian International Conference on Artificial Intelligence, India, pp. 465-472.

National Convention

1. Jain, R. and Chaudhari, N.S., (2011), Cluster Analysis in Social Networks & 2-CNF, Twenty Fifth National Convention of Computer Engineers and National Seminar on Networked Home Systems and Services, India, pp. 169-172.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

ACRONYMS

MSDC: Minimum Sum of Diameter Clustering

MSRC: Minimum Sum of Radii Clustering

NP-hard: Non-deterministic Polynomial Hard

NP-Complete: Non-deterministic Polynomial Complete

SAT: Satisfiability

2-SAT: 2-Satisfiability

3-SAT: 3-Satisfiability

k-SAT: *k*-Satisfiability

ML: Must Link

CL: Can-Not Link

CNF: Conjunctive Normal Form

PTAS: Polynomial Time Approximation Scheme

QPTAS: Quasi-Polynomial Time Approximation Scheme

TC: Time Complexity

AF: Approximation Factor

DMD: Distinct Maximal Disc

Chapter 1

Introduction

While pursuing any research, researchers have to deal with a wide variety of data emanating from all sorts of measurements and observations. A large quantity of data is being generated inform of documents, reports, e-mails, and web pages are generated from different sources, like as enterprises, governments, organizations, and individuals. Generation of data in huge quantity leads towards data analysis. Data describe the characteristics of a living species, depict the properties of a natural phenomenon, summarize the results of a scientific experiment, and record the dynamics of a running machinery system. More importantly, data provide a basis for further analysis, reasoning, decisions, and ultimately, for the understanding of all sorts of objects and phenomena. A systematic and automatic approach is imperative to organize this unstructured data without human intervention. Data analysis is considered as one of the most essential activities to classify or group data into a set of categories or clusters. Further, this has paved the path for cluster analysis which is basically concerned with resolving the multifaceted problems related to partitioning of entities and to explore constraint clustering algorithm for minimum sum of diameter and minimum sum of radii problem.

1.1 Background

Following are the points of discussions which are dealt to illustrate terminologies and definitions pertaining to the topic of present research.

1.1.1 Cluster analysis

Cluster analysis [1-2] is mainly concerned with the problem of partitioning a given set of entities into homogeneous and well-separated subsets called clusters. Cluster analysis is all about finding subset that are homogeneous and/or well separated. However, the process of clustering explores natural groupings and thereby represents a holistic overview of classes which are in-

form of collection of documents. Indubitably, in the area of artificial intelligence, this concept is popularly known as unsupervised machine learning. Cluster analysis [3-5] is a technique to explore the structure of data; it is the body of methods that help to describe data, detect useful hidden patterns and develops explanations from large amounts of data. Clustering is to identify classes of similar objects among a set of objects. Two objects are said to be similar if they exhibit a coherent pattern on a subset of dimensions. Cluster analysis is a key technique in the data analysis and is being applied in a variety of engineering and scientific disciplines such as biology, psychology, medicine, marketing, computer vision and remote sensing [6]. Clustering techniques have been used in a wide variety of application areas including information retrieval, image processing, pattern recognition, DNA microarray analysis and E-commerce applications [6–9].

Clustering algorithms partition the data into homogeneous and/or well separated classes called clusters. Two important concepts of cluster analysis are internal homogeneity and external separation [10-12]. Internal homogeneity means patterns in the same cluster should be similar to each other and external separation means patterns in different clusters should be different. One of the homogeneity measures is the diameter of the cluster. The diameter of a cluster is the maximum dissimilarity between any pair of entities in that cluster. Minimum diameter clustering is an important and traditional clustering method, but in some applications minimum diameter clustering algorithm propagates dissection effect [1,4,13]. Dissection effect causes similar objects that should be placed in the same cluster to be assigned to different clusters. Clusters tend to have fairly equal diameters and this may cause the dissection of some natural cluster. To avoid dissection effect, the minimum sum of the diameters or minimum sum of radii can be selected as a criterion.

1.1.2 Minimum sum of diameter clustering and minimum sum of radii clustering

Minimum sum of diameter clustering (MSDC) [4] is the partitioning of the entities such that the sum of the diameters of the clusters is minimized. Minimum sum of radii clustering (MSRC) [4] is the partitioning of the entities is like such that the sum of the radii of the clusters is minimized. MSDC and

MSRC problems belongs to class of NP-minimization problem [14]. Brucker [15] revealed the problem of determining a partition of a given set of N entities into k clusters as the sum of the diameters of these clusters is minimum is NP-complete for $k \geq 3$ and its complexity was unknown for $k = 2$.

1.1.3 Approximation clustering algorithms

Approximation algorithms [16] are applied to know and find approximate solutions for optimization problems. Approximation algorithms are usually linked with NP-hard [17] problems. An algorithm is considered to be a ϵ-approximate algorithm for a problem P, iff either posses

1) P is a maximization problem and for every instance of P,

$$\left|\frac{\mathrm{Opt}(P) - \mathrm{Apx}(P)}{\mathrm{Opt}(P)}\right| \leq \epsilon \tag{1}$$

Where, $0 < \epsilon < 1$

2) P is a minimization problem and for every instance of P,

$$\left|\frac{\mathrm{Opt}(P) - \mathrm{Apx}(P)}{\mathrm{Opt}(P)}\right| \leq \epsilon, \tag{2}$$

Where, $\epsilon > 0$

Opt(P) is the optimal solution (assuming Opt(P) > 0) and Apx(P) is the derived approximate solution.

1.1.4 Constraint clustering

Constraint clustering [18] is the most prominent area of machine learning and data mining-oriented research. Constraints facilitate hands on information about the desired partition and strengthen performance of clustering algorithms. The key function of clustering algorithms is not only to encompass all the domain expert's requirements but also instrumental in directing the algorithm to a desirable set partition by adopting user specified constraints whereas constraint clustering stimulates composition of a desirable clustering of the instances. Accuracy of clustering algorithms can be improved by clubbing such constraints.

1.1.5 Terminology

Definition 1 (*k*-clustering [19-20]): Let $O = \{O_1, O_2, \dots O_N\}$ denote a set of $N = |O|$entities and $D = \{d_{ij} | i \leq k, l \leq N, 1 \leq j \leq N\}$ a set of dissimilarities between pairs of these entities. Dissimilarity d_{ij} is a real number and satisfies to the conditions $d_{ij} \geq 0, d_{ii} = 0$ and $d_{ij} = d_{ji}$ for $i, j = 1,2, \dots N$. The *k*-clustering of *O* entities into *k* clusters $C = \{C_1, C_2, \dots C_k\}$ is such that no cluster is empty, any pair of clusters has an empty intersection and the union of all clusters is equal to *O*. Diameter of clusters $C_1, C_2, \dots, C_k$ are $r_1, r_2, \dots, r_k$respectively, assuming that $r_1 \geq r_2 \geq \cdots \geq r_k$ respectively.

Definition 2 (Diameter of Cluster [4, 10, 21]): The diameter of a cluster $d(C_l)$ is the maximum dissimilarity between any pair of entities in that cluster. In other words, it is the largest dissimilarity between entities in C_l, where $C_l \in C$.

$$d(C_l) = \max_{O_i, O_j \in C_l} d_{ij} \tag{3}$$

Definition 3 (Radii of Cluster [22]): The radii of a cluster $r(C_l)$ is the maximum dissimilarity of any point in the cluster from the cluster center. If O_i is the center of cluster C_l then radii of cluster is given by the formulation:

$$r(C_l) = \max_{O_i, O_j \in C_l} d_{ij} \tag{4}$$

Definition 4 (Minimum Sum of Diameter [4, 10]): Minimum sum of diameter clustering is the partitioning of the entities viz. the sum of the diameters of the clusters is minimized.

$$d(C_M) = \text{Min} \sum_{l=1}^{k} d\,(C_l) \tag{5}$$

Definition 5 (Minimum Sum of radii [22]): Minimum sum of radii clustering is the partitioning of the entities viz. the sum of the radii of the clusters is minimized.

$$d(r_M) = \text{Min} \sum_{l=1}^{k} r\,(C_l) \tag{6}$$

1.2 Problem Considered

The present research makes an attempt to address, study, analyze and resolve the problem as:

1.2.1 Constraint Clustering and *k*-Clustering to *k*-SAT

One of the important applications of cluster analysis is social network analysis [23]. In social networks, nodes of the network are people and the links are the relationships between people. Social network analysis practitioners collect network data, analyses the data and often produce maps or pictures that display the patterns of connections between the nodes of the network. These maps reveal characteristics of the clusters. Bonding and bridging are two different kinds of connectivity in social network. Concept of bonding and bridging are translated into Satisfiability (SAT) [24] formulation. In this problem, goal is to investigate ML-constraints and CL-constraints with the help of bonding and bridging. *k*-clustering problem is reduced into *k*-SAT formulation with the help of investigates constraints in the social networking.

1.2.2 Partitioning Problem

A partition [25] of a set U is a subdivision of the set into subsets that are disjoint and exhaustive, i.e. every element of U must belong to one and only one of the subsets. The subsets P_i in the partition are called cells. Thus $\{P_1, P_2, \ldots P_r\}$ is a partition of U if two conditions are satisfied: (1) $P_i \cap P_j = \emptyset$, if $i \neq j$ and (2) $P_1 \cup P_2 \cup \ldots \cup P_r = U$. In the field of Computer Science, the partition problem is an NP-complete problem [25-26] and it is also NP-hard [27] to find good approximate solutions for this problem. In 3-partitioning there are three partitions, $U = P_1 \cup P_2 \cup P_3$. In this research work, bit wise technique is used for the generation of partitions.

1.2.3 Minimum Sum of Diameter Problem for 3-clustering

Traditional clustering algorithms like *k*-clustering algorithm or minimum diameter clustering algorithm are highly affected by the dissection effect. In dissection effect, similar entities may be assigned to different clusters. To avoid this effect, it is proposed to have minimum sum of diameter clustering or minimum sum of radii clustering. Approximation algorithms exist for minimum sum of diameter for three clustering but still there is no exact algorithms is available. The present research investigates exact algorithm for minimum sum of diameter for three clustering problem.

3-clustering is partitioning of O into 3 clusters C_1, C_2, and C_3 such that no cluster is empty, any pair of clusters has an empty intersection and the union of all clusters is equal to O. Minimum sum of diameter for 3-clustering is partitioning of the entities into three clusters such that their sum of diameter of cluster is minimum. Mathematically it can be defined

$$min \sum_{i=1}^{3} d(C_i) \tag{7}$$

Where, $d(C_i)$ is diameter of cluster C_i.

1.2.4 Constraint Word Clustering Algorithm for Asymmetric Relationship

In this research work, a new constraint word clustering [28] algorithm is proposed for the given corpus. The proposed method is based on the constraint clustering of words. In this context, words are considered similar if they appear in similar contexts and contexts are similar if their word affinity clouds are equivalent. Different sorts of association among words are identified and constraints are identified and generated according to this association. Proposed constraint algorithm is applicable for words having asymmetric relationship between them; therefore this approach may be useful as a complement to the conventional class-based statistical language modeling techniques.

1.2.5 Constraint Clustering for Minimum Sum of Diameter and Radii Clustering Problem

Consider the min-cost *k*-cover [29] problem: For a given a set *P* of *n* points in the plane, objective is to cover the *n* points by *k* disks, such that sum of the radii of the disks is minimized. The concept of constraints is being introduced in min-cost *k*-cover problem to present a new constraint based min-cost *k*-cover algorithm. Investigations formulate that a can-not link constraint always separates the optimal solution and reduces cardinality of distinct maximal discs. In any instance of min-cost *k*-cover problem, upper bound and lower bound on the number of can-not link constraints are $O(n^2)$ and $O(k)$ respectively.

1.3 Motivation

The main motivation behind the present research is to find Boolean formulation for clustering problem and find constraint clustering algorithm for minimum sum of diameter and minimum sum of radii clustering algorithm. Review of literature also reveals that there is no constraint based clustering algorithm available in exact and approximate algorithms for minimum sum of diameter and radii clustering. All these elements encourage to pursue the present study and to develop a new dimension in the clustering algorithm by giving constraint clustering algorithm for minimum sum of diameter and radii. Motivation behind using for the present research work is to investigate algorithms available on minimum sum of diameter and minimum sum of radii, and prepare the literature survey on comparison of various techniques of minimum sum of diameter and radii clustering. Research also motivates to introduce the reduction techniques for 3-cluster to 3-SAT and *k*-cluster to *k*-SAT. Further, constraints based techniques are introduced in word clustering algorithms and in approximation algorithm for minimum sum of diameter.

1.4. Objectives and Scope

The main objective is to review and draw a comparative analysis of algorithmic complexity of various exact and approximation clustering algorithms for Euclidean, metric and geometric version of MSDC and MSRC problem in chronological order of their evolution. The above comparative analysis may open new vistas of knowledge in the field of clustering and

address the emerging challenges, relevant issues, innovative ideas, recent trends, advancements and future scope for research in MSDC and MSRC problem in relation with theoretical computer science. The research deals with investigation, identification of constraints and generation of the constraints and reducing them into SAT formulation. This implies standard clustering methods using constraints to obtain results. The research also aims to highlight minimum sum of diameter and minimum sum of radii clustering based on constraints.

The scope of the research is interwoven with situations in which homogeneity of the clusters is found with respect to natural grouping. It is applicable in routing, in application of location theory and in communication network design**.** An example of application of this theory is reflected in telecommunication establishment [30]: a network of base stations (antennas) such that all the locations are within the range of some station and sum of setup cost (proportional to the diameter/range of the station) of the station is minimized.

1.5 Research Gap

The prior sections have discussed about the existing techniques exclusively seen for addressing minimum sum of diameter and minimum sum of radii clustering. There are various forms of the clustering technique to group the data items. However, there are effectiveness as well as limitations associated with almost all the existing systems.

- The concept of reduction from 3-clustering to 3-SAT and *k*-cluster to *k*-SAT are still unaddressed. Existing reduction techniques focuses only from SAT formulation to clustering.
- Adequate literature survey is not available on minimum sum of diameter and radii clustering algorithms that identifies and compare the constraints, assumption, issues and complexity of algorithms.
- Existing word clustering algorithms are not involved in any type of pre-computation on similarity matrix. Word clustering techniques with pre-computation that select the input parameters as constraint to identify the semantic relationship is not addressed.

- Existing clustering algorithms still has an open problem to solve the minimum sum of diameter and radii in polynomial time. To reduce the time complexity many approximation algorithms are available in literature. The complexity of model selection as it is dependent on the cluster properties or relationship between the data items as input. Therefore, still now, much clarity is not illustrated how to use the constraint-based techniques for approximation algorithm for MSDC and MSRC in any research work.

1.6 Contribution of Research work

The research deals with conceptual framework in the field of computer science and has its focus on exploring application of constraint-based techniques related with cluster analysis. This constraint-based technique derives the essential characteristics of the data and facilitates analysis as well as algorithm design. The major contributions are as:

- The research study enables the future researchers to have a comprehensive and analyzed information about the application of constraint based clustering technique as prior to thesis work no adequate information was available to explore the minimum sum of diameter and minimum sum of radii clustering algorithm.
- The research analyzed, identified and generates constraints in the social network and reduced them into SAT formulation. In this part, Belonging approach is proposed for the reduction of 3-cluster and *k*-cluster. Reduction methodology is formulated to reduce 3-cluster problem into 3-SAT formulation and *k*-cluster problem into *k*-SAT formulation.
- The research proposed a bit representation mechanism for the representation of partitions and investigated a 3-partitioning algorithm based on it. It also investigated constraint based exact algorithm for three clustering for minimum sum of diameter problem.
- We investigated various kinds of association between words in a given corpus. It further throws light on how constraint clustering could improve the performance of word clustering algorithm for asymmetric relationship between words.

- The constraint based model in min-cost *k*-cover algorithm is introduced. The analytical results prove that constraint based min-cost *k*-cover algorithm is more time efficient than algorithm which is in existence. During the modeling exercise, salient features (constraints) that enhance our understanding on the bifurcation of the disks, and therefore, constraints are relevant for the design and analysis of efficient min-cost *k*-cover algorithm.

1.7 Thesis Organization

To cover vast area of cluster analysis and to explore constraints based techniques related with minimum sum of diameter and minimum sum of radii clustering, the present research is divided into following chapters for better conceptual and analytical understanding of the selected research problem.

Chapter 1 deals background information on the minimum sum of diameter and minimum sum of radii clustering algorithm. Chapter 2 is literature review, based on previous research carried out in the domain of minimum sum of diameter and minimum sum of radii clustering algorithm. Chapter 3 describes how the bridging and bonding concepts of social network are transformed into SAT formulation. Reduction methodology highlights how 3-cluster problem is reduces into 3-SAT formulation and *k*-cluster problem is reduces into *k*-SAT formulation. Chapter 4 focuses on partitioning problem and exhibits a detailed description and experimental results on Bit partitioning method. Bit partitioning method is extended to represent exact 3-clustering algorithm for minimum sum of diameter clustering. Chapter 5 conveys different types of associations exist between words in a corpus. Architecture model for word clustering is discusses and then represented a word clustering algorithm for asymmetric relationship between words. Chapter 6 unfolds the importance of constraints in min-cost *k*-cover problem. Chapter 7 draws conclusion of research work and suggests future scope for research.

Chapter 2

Literature Review

2.1 Introduction

Cluster Analysis is one of the fields of research that falls under the sub-discipline of data analysis. The main reason to apply efforts in this field to analyze the vast data generated from reports, e-mails, documents, and web pages. The grouping of data provides a basis for analysis, reasoning, taking decisions. Clustering algorithms groups the data into separated classes, but due to dissection effect similar objects that should be placed in the same cluster to be assigned to different clusters. Minimum sum of diameter clustering is the partitioning of the entities such that the sum of the diameters of the clusters is minimized. This chapter covers a study of existing research work conducted on the minimum sum of diameter and minimum sum of radii clustering. An attempt has been made to have glimpse of the entire work done on this subject matter across the world. It was observed that there has not been much of literature review conducted on the subject matter minimum sum of diameter and minimum sum of radii clustering.

2.2 Satisfiability Problem

A Boolean expression is comprised of variables, parenthesis and the operators. A formula is in conjunctive normal form (CNF) if a Boolean expression is represented as conjunction of disjunctions, where each disjunction has two arguments that may either be variables or the negations of variables. The conjunctive normal form of formula is as: $(a \vee b) \wedge (b \vee c) \wedge (c \vee a)$. An expression is satisfiable if there are some assignments of 0's and 1's to the variables that demonstrated the expression of the value 1. The satisfiability problem [31, 32] is to determine a Boolean expression, whether it is satisfiable. 2-satisfiability (2-SAT) [32] is the problem of determining the satisfiability of a formula in conjunctive normal form where each clause is delimited to at most

two literals. When the clause size is greater than two, the problem is known as NP-complete. The Cook-Levin theorem [24] states that the Boolean satisfiability problem is basically NP-complete. 2-SAT problem is solvable in linear time [33]. 3-satisfiability (3-SAT) is the problem of determining the satisfiability of a formula in conjunctive normal form where each clause is limited to at most three literals.

2.2.1 Skew Symmetric Directed Graph

In implicative normal form [34] each disjunction is replaced by one of the two implications to which it is equivalent. Disjunction $X_1 \vee X_2$ is replaced by: $\bar{X}_1 \longrightarrow \bar{X}_2$ and $X_1 \longrightarrow X_2$. The implicative normal form of a 2-satisfiability problem can be represented as an implication graph. A skew-symmetric directed graph [34] has one vertex per variable or negated variable. An edge connects one vertex to another whenever the corresponding variables are related to an implication in the implicative normal form of the instance. If there is implication rule $X_1 \longrightarrow X_2$ then an edge goes from a vertex X_1 to vertex X_2.

2.2.2 Strongly Connected Component

Let $G = (V, E)$ be a directed graph. $V'$is the set of vertices belong to V. E' is the set of edges connecting pairs of vertices in V'. If v and w belong to V' and If there is a path from v to w and a path from w to v then the graph $G = (V, E)$ is called strongly connected components [34-35] of G. A graph is called strongly connected if there is a path from any vertex to any other. The maximal strongly connected subgraphs of any graph are vertex-disjoint and are called its strong components of it. Algorithm 2.1 produces strongly connected component of given graph *G* in linear time.

2.2.3 Algorithm for Strongly Connected Components

Data Structures used:

1. Stack: to store the visited nodes.
2. Oldnode: an array that contain visited nodes.
3. Newnode: an array that contains unvisited nodes.

4. Depth First Number (DFN): a single dimensional array to hold the depth first number.
5. Low Link (LL): a single dimensional array to hold the low link value of node. Initial value for all nodes is zero.
6. Count: a counter variable that has initial value of 1.

Algorithm 2.1 Strongly Connected Component SCC(*G*)

Input: A directed graph $G = (V, E)$, adjacency list L
Output: A list of the strongly connected components of G

1. count = 1
2. **for** all *v* in *V* do mark *v* "new"
3. Initialize STACK to empty
4. **while** there exists a vertex *v* marked "new" **do**
5. Search(*v*)
6. **end while**
7. **end for**

Subroutine: Search(*v*)

Input: Vertex *v*
Output: Vertex of strongly connected component

1. Mark *v* "old"
2. DFN[*v*] = count
3. count = count + 1
4. LL[*v*] = DFN [*v*]
5. push (*v*)
6. **for** each vertex *w* on L[*v*] **do**
7. **if (***w* is marked "new") **then**
8. search(*w*)
9. LL[*v*] = min (LL[*v*], LL[*w*])
10. **else**
11. **if** ((DFN[*w*] < DFN [*v*]) and OnStack (*w*)) **then**
12. LL[*v*] = min (DFN [*w*], LL[*v*])
13. **end if**
14. **end for**
15. **if** (LL[*v*] = DFN[*v*]) **then**
16. **repeat**
17. pop *x* from the top of stack;

18. print *x*
19. **until** *x* = *v*;
20. **end if**

Description of Algorithm:

As we know that the vertices of each strongly connected component are a connected subgraph of the spanning forest determined by the depth-first search. This connected subgraph is a tree and the root of the tree is called the root of the strongly connected component. Depth-first search is used to find the strongly connected components of a graph. To find out the strongly connected component a function LOWLINK is defined in the following manner:

LOWLINK[*v*] = MIN({*v*} U {*w* | there is a cross edge or back edge from a descendant of *v* to *w*. and the root of the strongly connected component containing w is an ancestor of *v*}).

The LOWLINK computation occurs at lines 4, 9, and 12. At line 4 LOWLINK[*v*] is initialized to the depth-first number of vertex *v*. At line 9 LOWLINK[*v*] is set to LOWLINK[*w*]. if for some son *w*, LOWLINK[*w*] is less than the current value of LOWLINK[*v*]. At line 11 we determine whether (*v*, *w*) is either a back edge or cross edge and we check to see whether the strongly connected component containing *w* has been found. If not, then the root of the strongly connected component containing *w* is an ancestor of *v*.

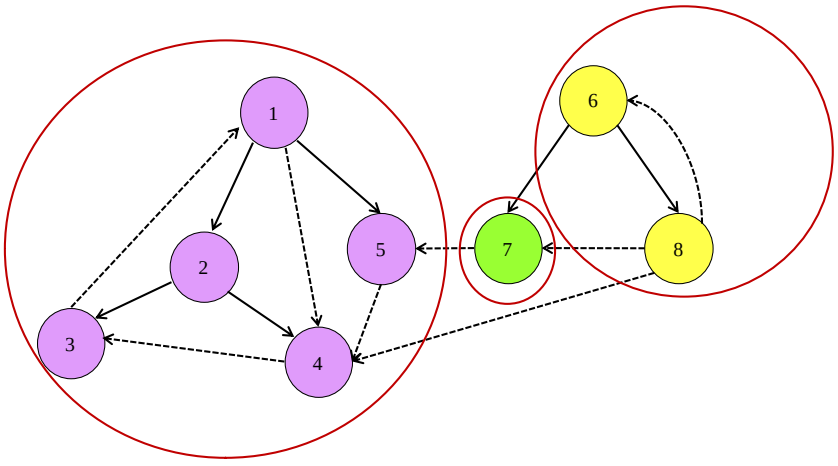

Figure 2.1: Strongly connected component of graph *G*

At line 12 we set LOWLINK[v] to the depth-first number of w. If it does not already have a lower value. When all of the vertices of a node are discovered and LOWLINK [v]= DFN[v] then elements from the stack are popped out. Stack is emptied and the list of the vertices belong to strongly connected component of G. This procedure is performed for all vertices of G. Applying the Algorithm 2.1 on the Graph G in Figure 2.1 produces three strongly connected component (SCC).

S_1 = {1, 2, 3, 4, 5} S_2 = {7} S_3 = {6, 8}

2.2.4 Linear Time Algorithm for 2-SAT Problem

Aspvall *et al*. in 1979 [33] presented a linear time algorithm for testing the truth of certain quantified Boolean formulas.

2.2.4.1 Procedure to Check Solvability of 2-SAT Problem

Algorithm in [33] uses properties of directed graphs. Suppose we are given a formula $F = Q_1 x_1 Q_2 x_2 \ldots\ldots Q_n x_n C$ such that C is in conjunctive normal form with at most $G(F)$ with $2n$ vertices and $2|C|$. If we assign truth values to the vertices of $G(F)$, Such an assignment corresponds to a set of truth values for the variables which makes C true if and only if:

(i) For all i, vertices x_i and $\bar{x}_i$ receive complementary truth values.

(ii) No edge $u \rightarrow v$ has u assigned true and v assigned false (equivalently, no path leads from a true vertex to a false vertex)

Steps of Checking Solvability of 2-SAT Problem:

a. Find out implicative normal form of disjunction of all pair of variables from the conjunctive normal form.
b. Draw implication graph and a skew-symmetric directed graph from implicative normal form where the vertices are the set of literals from a 2-CNF formula.
c. Explore the strongly connected component in the implication graph.
d. Apply 2-satisfiability algorithm to process strongly connected components.
e. A formula is satisfiable if, and only if, no pair of literals, a and $\bar{a}$, appear in the same strongly connected component.

Theorem 1. The formula F is true if and only if none of the following three conditions holds:

(i) An existential vertex u is in the same strong component as its complement $\bar{u}$.

(ii) A universal vertex u_i is in the same strong component as an existential vertex $u_j$such that $j < i$ (i.e., x_i is not quantified within the scope of Q_i).

(iii) There is a path from a universal vertex u to another universal vertex v. (This condition consists the case that, $v = \bar{u}$).

2.2.4.2 Steps in 2-satisfiability Procedure

Step 1. If S is marked then move forwards to the next component. Otherwise if some successor of S is marked false or contingent lead to Step 2. Otherwise reach to Step 3.

Step 2. (S has a false or contingent successor.) If S contains one or more universal vertices, stop: Theorem 1 condition (iii) holds. Otherwise, mark S false and move to Step 5.

Step 3. (All successors of S are true.) If S contains two or more universal vertices, stop: Theorem 2 condition (iii) holds. Otherwise, if S contains one universal variable u_i go ahead with Step 4. Otherwise, mark S true and go to Step 5.

Step 4. (S contains a universal vertex u_i) If S contains an existential vertex u_j with $j < i$, stop: Theorem 1 condition (ii) holds. Otherwise, mark S contingent and directs to Step 5.

Step 5. (S is marked successfully.) If $S = S'$ stop: Theorem 1 condition (i) or (iii) holds. Otherwise, skip to Step 6.

Step 6. $S = S'$ If S is marked contingent or false and S' is a predecessor of S, stop: condition Theorem 1 (iii) holds. Otherwise, mark S' false if S is true, contingent if S is contingent and true if S is false; march towards the next component.

2.2.4.3. Description of 2-satisfiability Algorithm

This algorithm marks each component processed true, false, or contingent. Each component has a universal vertex which is marked contingent; each component containing only existential variables which is marked either true or

false. When a component is marked false, either all its successors are marked, at least one of them contingent or false, or all its predecessors are marked, all of them are false. It can be easily concluded from Step 2, Step 3, and Step 6 and the duality property. During the operation of the algorithm, some component S_1 is marked false while it has an unmarked predecessor, and then there is a path from S_1 to a component *S*, marked contingent. Similarly, when a component is marked true, either all its successors are marked, all true, or all its predecessors are marked, at least one true or contingent. Thus, if some component S_2 is marked true while it has an unmarked successor, then there is a path from some contingent component S_1 to S_2. It follows from these facts that if the algorithm stops at Step 2 or Step 6, then condition Theorem 1 (iii) holds. If the algorithm stops in Step 3, Step 4, or Step 5, it is obvious that the indicated condition holds. Thus, if the algorithm stops prematurely, at least one of the Theorem 1 conditions (i)-(iii) holds. If the algorithm does not stop prematurely, every component is marked in order to have true or contingent component has only true components as successors. Similarly, any false or contingent component has only false components as predecessors. This follows easily from the operation of the algorithm and the duality property. Furthermore, every component and its complement receive complementary truth values, and every contingent component has a universal vertex u_i that contains as additional vertices which is only existential vertices u_i such $i < j$. We can prove that *F* is true as follows: to each vertex in a true or false component, assigned value is true or false, respectively. For any assignment of truth values to the universal variables, we assign to each vertex in a component containing a universal vertex U_i the truth value of x_i if $u_i = x_i$ and the complementary truth value if $u_i = \bar{x}_i$. Thus, in this context *F* is true.

2.2.4.4 Complexity of 2-satisfiability Algorithm

2-satisfiability algorithm requires $O(n + m)$ time, where *m* is the number of edges in G(*F*) (twice the number of clauses in *C*). The algorithm processes strong components in the same order as they are generated by the linear-time strong components algorithm; thus, strong components algorithm may be used with only minor modifications as to solve this evaluation problem.

2.3 Literature Survey of 2-clustering Algorithms

In the following sub points, a great impetus is given on the evolution of NP-complete problems, polynomial time reduction of NP-complete problems and formulation of cluster analysis as mathematical programming. Various 2-clustering algorithms are also analyzed on the basis of time complexity.

2.3.1 NP-complete Problems and Polynomial-time Reducibility of Problems

Computational problems are quite complex and stimulating hence they have to be resolved at the top priority, efficiently and successfully to settle the issue for once and all. But contrary to this still there are some problems that are hard to be resolved and hence known as NP-hard problems. Computational complexity theory is a branch of theoretical computer science which focuses on classifying computational problems. In computational complexity theory, there are lots of computational problems which can be solved by a non-deterministic Turing machine [14] in polynomial time, such a problem is treated as non-deterministic polynomial (NP) problem [34, 36]. NP problems are the set of decision problem [25]. Foundation of NP-complete (non-deterministic polynomial complete) problems and the notion of polynomial-time reducibility is investigated by Cook [24]. If a polynomial time reduction algorithm exists for reducing one problem to another then this means that any polynomial time algorithm for the second problem can be converted into a corresponding polynomial time algorithm for the first problem. Further it is proved existences of a NP-complete problem by showing that the Boolean satisfiability problem (SAT) is NP-complete. Karp [17] demonstrated set of 21 computational problems which are NP-complete.

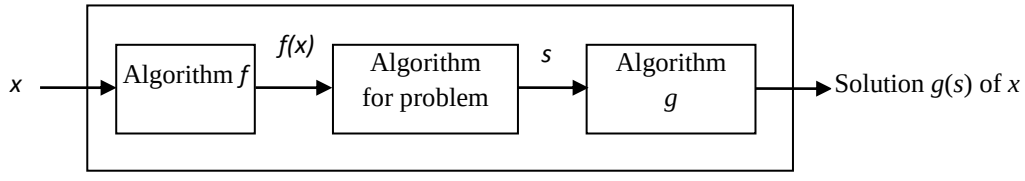

Figure 2.2: Algorithm for problem *A*

It can be inferred that there is a polynomial time many-one reduction of the Boolean satisfiability problem to each of the 21 combinatorial and graph

theoretical computational problems. It means that many natural computational problems are computationally intractable [24]. A large number of computational problems when expressed as language recognition problems, arising in fields such as mathematical programming, graph theory, combinatorics, computational logic and switching theory are NP-complete.

Theorem 2. [24] If a set *S* of strings is accepted by some nondeterministic Turing machine within polynomial time, then *S* is *P*-reducible to disjunctive normal form tautologies.

According to Theorem 2, any recognition problem can be solved by a polynomial time-bounded nondeterministic turing machine that can be reduced to the problem of determining whether a propositional formula is a tautology or not. Polynomial time reduction [37] implies that first problem can be solved deterministically in polynomial time provided a solution which is available for solving the second problem. From Figure 2.2 it is clear that if problem *B* can be used to solve *A*, then problem *A* can be reduced to problem *B*. A problem is polynomial time reducible: if a polynomial time algorithm f transforms any instance x of the problem *A* into an instance $f(x)$, together with another polynomial time algorithm g that maps any solution s of $f(x)$ back into a solution $f(x)$ of x. If algorithm f and algorithm g are efficiently computable, then this develops an efficient algorithm for the problem *A* to reduce into problem *B*.

2.3.2 Formulation of Cluster Analysis as Mathematical Programming

Rao [19] explored the distance base cluster analysis and focused on the problem of optimal partitioning and showed how the problem can be formulated as a mathematical programming problem. Problem formulation on the basis of different distance-based criterion are as follows:

1) *Minimize the within group's sums of squares distance*: In this formulation an efficient dynamic programming algorithm is provided, considering entities are points on the real line.

$$\text{Min} \sum_{i=1}^{N} \sum_{j=1}^{M} w_i x_{ij} \left(q_i - \bar{q}_j\right)^2 \tag{8}$$

where, $\bar{q}_j = \left(\sum_{i=1}^{N} w_i x_i q_i\right) \Big/ \sum_{i=1}^{N} w_i x_i$

$$\sum_{j=1}^{M} x_{ij} = 1 \qquad i = 1, 2, \ldots, N$$

where, $x_{ij} = 0 \; or \; 1, \;\; \forall \; i, j$

W_i is the weight assigned to entity i, q_i is a measure assigned to entity i. x_{ij} = 1 or 0 depending on whether entity i is assigned to group j.

2) *Minimize the sum of average within group squared distances*: Two approaches are given to solve the problem: In the first approach, problem is treated as a constrained non-linear Boolean programming problem, this approach followed the method of Hammer et al. [38]. In the second approach, objective function was to linearize at the cost of increasing the number of constraints and then solving the resulting problem by adopting linear integer programming technique.

$$\text{Min} \sum_{k=1}^{N} \left[\frac{\left(\sum_{j=1}^{N-1} \sum_{j=i+1}^{N} d_{ij}^2 x_{ik} x_{jk} \right)}{\sum_{i=1}^{N} x_{ik}} \right] \tag{9}$$

Subjected to:

$$\sum_{k=1}^{M} x_{ik} = 1$$

where, $x_{ik} > 0$, $i = 1, 2, \ldots, N$ and $k = 1, 2, \ldots, N$.

3)*Minimize the total within group distance*: In this formulation, objective function of criterion 2 can be reduced into the objective function of criterion 3, then linearizing the new objective function and solving the resulting problem by applying linear integer programming technique.

$$\text{Min} \sum_{k=1}^{M} \left(\sum_{i=1}^{N-1} \sum_{j=i+1}^{N} d_{ij} x_{ik} x_{jk} \right) \tag{10}$$

$$\sum_{i=1}^{N} x_{ik} = n_k \quad k = 1,2, \dots, M \tag{11}$$

where, n_k is the number of entities in group k such that

$$\sum_{k=1}^{M} n_k = N$$

4)*Minimize the maximum within group distance*: A linear integer programming solution is modeled for minimize the maximum within group distance. In this solution approach, the number of constraints increases incessantly with N (number of entities) and k (number of clusters) and therefore, this formulation is computationally useful only for small values of N and k.

Min Z,

Subjected to:

$$d_{ij}x_{ik} + d_{ij}x_{jk} - Z \leq d_{ij} \tag{12}$$

where, $i = 1, 2, \dots, N\text{-}1$, $j = i+1, i+2, \dots, N$ and $k = 1, 2, \dots, M$

such that $\sum_{k=1}^{M} x_{ik} = 1$ $\quad i = 1, 2, \dots, N$

x_{ij}, $Z > 0$ and integer valued

2.3.3 Minimum Sum of Diameters as NP-complete Problem

Brucker [15] put forth a problem of partitioning a given set of N entities into k clusters, such that the sum of the diameters of these clusters is minimum as an NP-complete problem for $k \geq 3$ and even has unknown complexity for $k = 2$. The NP-completeness of MSDC problem for $k \geq 3$ is an open problem for the researchers. When sum of diameter as other than as criterion is chosen, clustering algorithms have constraints on the number of entities that can be clustered. Hansen *et al*. [20] examined a complete-link cluster analysis algorithm for minimization of maximum dissimilarity between entities in the same cluster but it is delimited up to 270 entities. Similarly, Delattre *et al*. [13] solved clustering problem in which both homogeneity and separation of the

clusters are simultaneously considered. Their research focused on the concept of split and diameter. A single-link algorithm may have a chaining effect [39] because of the poor homogeneity of clusters and the complete-link algorithm may have dissection effect due to badly separated clusters. This problem is solved by using bi-criterion cluster analysis taking both diameter and split criteria simultaneously into account to obtain partitions without such defects. By considering both criterions simultaneously their algorithm ensures maximum homogeneity and maximum separation. Their algorithm is also having bound on the number of entities and can determine a minimal complete set of efficient partitions with up to 400 entities.

2.3.4 Linear Time Algorithm for 2-SAT Formulation

Aspvall *et al.* [33] redefined a linear-time algorithm for the evaluation of a 2-satisfiabile formula. The algorithm processes the Boolean formula and constructs a directed graph from the quantified Boolean formula of *n* variables. In the first step of this algorithm, a directed graph developed on the basis of Boolean formula. In the second step of this algorithm, a linear time algorithm [35] is found as a strongly connected component. Time complexity of linear time algorithm is $O(\max(n, m))$, where *n* is the number of vertices and *m* is the number of edges. Even *et al.* [40] dealt with a linear-time algorithm. Schaefer *et al.* [41] also worked on efficient polynomial time solution for conjunctive normal formula having only two literals per clause. Their research proposed a simple constructive algorithm for the evaluation of formulas having two literals per clause, which runs in linear time on a random-access machine. This algorithm is based upon the Aho *et al.* [35] and Tarjan *et al.* [42] algorithm, which talked about strong components of a directed graph in linear time. It requires $O(n + m)$ time, where n is the number of variables in a Boolean formula and *m* is twice the number of clauses in a Boolean formula.

2.3.5 2-clustering Algorithms for Minimum Sum of Diameter and Minimizing the Maximum diameter

Under this head, various 2-clustering algorithms for the minimum sum of diameter and minimization of largest diameter are dealt. Let *N* is the number of entities in a complete graph *G*, C_1 and C_2 are clusters and D_1 and D_2 are their

diameter respectively. 2-clustering problem is finding a partition of set O of N entities into two non-empty clusters C_1 and C_2, such that the sum of diameter of cluster C_1 and cluster C_2 is minimized. Rao [19] researched minimizing the maximum diameter clustering problem as mathematical programming through linear programming. After this research finding, a huge research gap is witnessed due to non-availability of related review of literature. A comparative investigation of approach, constraints, issues, running time computation of various exact 2-clustering algorithms for minimizing the sum of diameter and minimizing the maximum diameter problem is exhibited in Table 2.1.

2.3.5.1 2-clustering algorithm based on 2-SAT Formulation

2-clustering for minimum sum of diameter is a polynomial time algorithm. Complexity of Algorithm 2.2 is $O(N^3 logN)$. For three or more clusters obtaining a minimum sum of diameters partition is an NP-complete problem. This 2-clustering algorithm is based on reduction of 2-clustering problem into 2-SAT formulation.

Theorem 3. Minimum sum of diameters bipartition can be solved in $O(N^2)$ time by reduction to the determination of the consistency of a quadratic Boolean equation.

Proof: Reduction of 2-clustering problem to 2-SAT formula for the bipartition $\{C_1, C_2\}$ of O is assigned by associating a Boolean variable x_i to each entity of O such that:

$$x_i = \begin{cases} 0, & if\ O_i \in C_1 \\ 1, & if\ O_i \in C_2 \end{cases} \tag{13}$$

Assuming,$r_1 \geq r_2$, Considering a pair $\{O_k, O_l\}$ of entities of O; there are three possibilities:

Case 1: If $d_{kl} > r_1$, then O_k and O_l both cannot belong to the same cluster C_1 or C_2, so the following boolean condition holds:

$$x_k x_l \vee \bar{x}_k \bar{x}_l = 0 \tag{14}$$

Case 2: If $r_1 \geq d_{kl} > r_2$, then O_k and O_l both cannot belong to the same cluster C_2, so the following boolean condition holds:

$$x_k x_l = 0 \tag{15}$$

Case 3: If $r_2 \geq d_{kl}$, then there is no restriction on whether O_k and O_l both should or should not belong to the cluster C_1 or C_2.

After analyzing the binary relation for each pair $\{O_k, O_l\}$ of entities, implied by the value of d_{kl} with respect to r_1 and r_2, formulates a 2-SAT formula.

2-clustering Algorithm Based on Reduction to 2-SAT Formula

Algorithm 2.2 2-clustering Algorithm for MSD

Input: Graph *G*

Output: Cluster C_1 and Cluster C_2 such that sum of diameter is minimized

1. Build maximum spanning tree to find d_{min} (first odd cycle in the spanning tree)
2. Identify the set of all candidate edge d_1, which are possible candidates for D_1.
3. For each candidate d_1 in D_1: Identify the value of d_2 in binary search manner such that there exists a partitioning of the cluster graph into two sets with diameters not exceeding d_1 and d_2 respectively.
4. For each entity x_i of the bipartition associate a Boolean variable
5. Apply Boolean approach to construct a 2-SAT expression *E* for d_1 and d_2

$$x_i = \begin{cases} 0, & if\ O_i \in C_1 \\ 1, & if\ O_i \in C_2 \end{cases}$$

6. Check satisfiability of the 2-SAT expression *E* using algorithm from [33]
7. If *E* is satisfiable then search for a lower value of d_2 else search for a higher value of d_2.
8. Choose D_1 and D_2 to pair (d_1, d_2) such that the sum of d_1 and d_2 is minimized.

2.3.5.2 2-clustering Algorithm for Minimizing the Largest Diameter

Asano *et al.* [43] presented an algorithm which partitions a set *S* of *n* points into two subsets so that their largest diameter is minimized in time $O(n\log n)$. In this algorithm optimal partition is determined by either a maximum or minimum spanning tree. Monma *et al.* [44] illustrated an $O(n^2)$ algorithm for the minimum sum of diameters two clustering. Guénoche *et al.* [21] presented a divisive hierarchical clustering algorithm for selecting the cluster with the largest diameter and partitioning it into two clusters whose largest diameter is possibly smallest one. They revealed two such algorithms: first algorithm is

just an efficient implementation of Hubert [45] algorithm that find all partitions into at most M clusters in $O(kn^2)$. For a fixed value of k the time complexity of the algorithm is $O(n^2)$. Second algorithm is a modified version of Rao [19] algorithm, which allows to build a complete hierarchy of partitions in $O(n^2 \log n)$ time.

2.3.5.3 2-clustering Algorithm for MSDC Based on Partitioning of Set

Hershberger [46] presented an algorithm for partitioning a planar set S of n points into two subsets so that the sum of the diameters of the subsets is minimized that is having bound of $O\left(\frac{nLog^2 n}{loglogn}\right)$.

Theorem 4. [46] *Given a planar set S of n points, the bipartition* $S = S_1 \cup S_2$, *that minimizes* $Diam(S_1)+Diam(S_2)$ *can be found in* $O(n\log^2 n)$ *time.*

Algorithm 2.3 PlanarPartition(*S*)

Input: Planar Set *S* of *n* Points
Output: Two subsets with minimized sum of the diameters

1. **for** *i* =l to *k* **do**
2. Merge the points in the radial interval $(r_{i-1}, r_i]$ into the voronoi diagram structure
3. **for** each point *p* in $(S_{i-1}, S_i]$ in radial order **do**
   3.1 Compute the farthest neighbor of *p* in the Voronoi diagram structure; call it
   3.2 Update the diameter $Diam(S_a) = \max\big(Diam(S_a), d(p,q)\big)$.
4. **end for**
5. **end for**

Algorithm 2.3 apply the concept of uses Union-Find algorithm [47] and use the concept of interval to identify the values of r when new points are added to the list. Merging of list and bi-partitioning is performed by applying Theorem 4. Any new point in the sector M, farthest point in the current set S is a or is in T or B; it cannot lie in M as shown in Figure 2.3. Any point in T or B is indicated that the algorithm doesn't need to check diameters for a substantial range of values of r. If a point q lies in T or B at a distance α from a, then no value of r in the following range can give a good partition: $(\sqrt{3}\alpha/2) \leq r < \alpha$

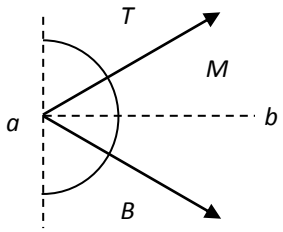

Figure 2.3: Partitioning of the plane into sectors

2.3.5.4 Challenge in the NP-complete Problems

Zuckerman [48] proved that all original 21 NP-complete problems have a version that is hard to approximate and these problems are meaninglessly hard to approximate. In fact, one cannot even approximate $\log^{(k)}$ of the magnitude of these problems to within a constant factor. 3-partitioning is one of the NP-complete problems from the list of 21 NP-complete problems. Hagauer *et al.* [49] presented an algorithm of $O(n^2\log^2 n)$ for finding a 3-clustering which minimizes the maximum diameter. This algorithm is based on the concept of Capoyleas *et al.* [50], in which any two clusters in an optimal solution can be separated by a line.

2.3.5.5 2-clustering Algorithm for MSDC Based on Look-ahead

Ramnath [51] presented algorithms that dynamically solve 2-SAT instances. Algorithm 2.4 dynamically maintains the transitive closure. The input can be represented by a weighted graph, Vertex i represented by item a_i, edge e_{ij} represents length l_{ij} between vertex i and vertex j. The output is a partitioning of the vertex set into two clusters C_0 and C_1, with diameters D_0 and D_1 respectively. Another algorithm dynamically maintains the partitioning of a graph into strongly connected components and runs in $O(n^3)$ and $O(mn)$ respectively, where m is the number of edges in the graph, n is the number of vertices. Both algorithms use the notion of perfect deletion lookahead which improves the time bound in comparison to notion of partial lookahead [52]. Partial lookahead is used to maintain the transitive closure that takes update times of $O(n^{2.18})$ with $n^{0.18}$ lookahead. Nomain *et al.* [53] illustrated that Ramnath [51] algorithm consistently performed better than P. Hansen *et al.* [4] algorithm.

Algorithm 2.4 2-ClusteringTransClos(G)

Input: Weighted graph G

Output: Two clusters C_0 and C_1

1. Identify all edge lengths, d_0, that is possible candidates for D_0.
2. Formulate the Type 0 constraint and Type 1 constraints for each edge e_{ij}.
3. Construct Type 0 constraint: If edge length(e_{ij}) > d_1, then set it as conjunct $(\bar{x}_i \vee \bar{x}_j)$.
4. Construct Type 1 constraint: If edge length(e_{ij}) > d_0 then set it as conjunct $(x_i \vee x_j)$
5. Insert all the Type 1 constraint into the constraint graph as undeletable edges.
6. Check constraint graph is satisfiable or not, by applying transitive closure or by decomposition it into strongly connected components. If it is not then delete the edges until the constraint graph is satisfiable.
7. Record the pair (d_1, d_2) such that the sum of d_1 and d_2 is minimized.

2.3.5.6 Discussion

In the context of minimum sum of diameter problem, 2-SAT formulation is solvable in polynomial; therefore 2-clustering problem is solvable in polynomial. To rationalized that 3-clustering problem is NP-complete problem, a methodology is proposed in [54] which reduced 3-clustering problem into 3-SAT formulation, 3-SAT formulation is NP-complete problem; therefore 3-clustering problem is also NP-complete problem. To rationalize that *k*-clustering problem is NP-complete problem; a methodology is proposed in [54] which reduced *k*-clustering problem into *k*-SAT formulation, *k*-SAT formulation is NP-complete problem; hence *k*-clustering problem is also NP-complete problem for minimum sum of diameter. 3-clustering problem can be solved in polynomial time, but it requires an exploration of properties of tripartite graph.

2.4 Literature Survey of Approximation Clustering Algorithm for Minimum Sum of Diameter and Radii Clustering Problem

Literature survey of approximation clustering algorithms for minimum sum of diameter and radii clustering problem is divided into two sections. Section first dealt with evolution of approximation algorithms, approximation algorithms for NP-complete problem, approximation algorithms to minimize the maximum diameter/radii of cluster and approximation algorithms for *p*-center

problem [16]. In Section second, approximation clustering algorithms for Euclidean, metric and geometric version of MSDC and MSRC problem are analyzed. A comparative investigation of approach, issues, constraints, and running time computation complexity of various approximation algorithms of MSDC problem, MSRC problem and related problems is shown in Table 2.2

2.4.1 Fundamentals of Approximation Algorithms

Approximation algorithms are applied to find approximate solutions to optimization problems. Approximation algorithms are mostly linked with NP-hard problems. Following are the definitions of the selected terminologies:

Definition 6 (Polynomial Time Approximation Scheme [55]). A Polynomial Time Approximation Scheme (PTAS) is a (1+ϵ) approximation algorithm that runs in time polynomial in the size of input but it can be exponential in 1/ ϵ. The running time of an PTAS is polynomial in n for every fixed ϵ but can be different for different value of ϵ. Thus, running time of an PTAS algorithm can be $O(n^{1/\varepsilon})$.

Definition 7 (Quasi-polynomial Time Approximation Scheme [55]). A Quasi-polynomial time approximation scheme (QPTAS) algorithm runs in time $O(n^{log^c n})$ for a constant *c*.

Definition 8 (*p*-center Problem [56-60]). The *p*-center problem is to find out the optimal location of *p* nodes so that the maximum distance of a node to its nearest center is minimized. *P*-center problem is NP-hard even in the case of planar Euclidean.

2.4.2 Approximation Algorithms on Minimizing Diameter of Cluster and Minimizing Radii of the Cluster

In this sub section, approximation algorithms are analyzed on the basis of minimizing the diameter of cluster and minimizing the radii of the clusters.

.

Table 2.1: 2-clustering algorithms and minimizing the maximum diameter clustering algorithms: A Comparative Overview

Year	Authors Name	2-clustering Algorithm Criteria	Approach/Concept	Constraint/Assumption/Issues	Time Complexity
1987	Hansen *et al*. [4]	Minimizing the sum of diameter	Reduction of 2-clustering to 2-satisfiability (2-SAT)	Can't be extended for k >2	$O(n^3 \log n)$
1988	Asano *et al*. [43]	Minimizing the maximum diameter Maximizing the minimum intercluster distance	Maximum spanning tree Minimum spanning tree	For planar set, if convex hull is disjoint For convex polygon and vertices should be ordered	$O(n\log n)$ $O(n\log n)$
1989	Monma *et al*. [44]	Minimizing the sum of diameter	Bicoloring vertices of Maximum spanning tree	Algorithm applicable only for Euclidean plane with $O(n)$ space	$O(n^2)$
1991	Guénoche *et al*. [21]	Minimizing the maximum diameter	Optimal bicolorations of vertices	Applicable only for fixed value of k	$O(n^2)$
1992	Hershberger [46]	Minimizing the sum of diameter Minimizing the sum of diameter	Finding good bipartition and farthest-point voronoi diagram Merge the points in the radial interval into the voronoi diagram structure	Points should lie in a plane Ratio between diameter & minimum inter-point distance should be polynomial in n	$O(n\log^2 n/\log\log n)$ $O(n\log n)$
2002	Ramnath [51]	Minimizing the sum of diameter Minimizing the sum of diameter	Dynamically solving the 2-SAT instances and dynamically maintaining the transitive closure Dynamically solving the 2-SAT instances and dynamically maintaining the partitioning of a graph into strongly connected graph	Applicable only for 2-SAT instances, for 3 or more clusters number of constraints increase exponentially Applicable only for 2-SAT instances, for 3 or more clusters number of constraints increase exponentially	$O(n^3)$ $O(mn)$

2.4.2.1 2-approximation Algorithm for Minimizing the Maximum Intercluster Distance (1985)

Gonzalez [61] examined an 2-approximation algorithm to minimize the maximum intercluster distance that runs in $O(kn)$ time for m dimension, where n is the number of points, k is the number of clusters and m represents dimensionality (for all value of m). D.S. Hochbaum *et al.* [62] obtained same results but their algorithm is not applicable for $m > 1$. Their approximation algorithm proves successful as long as the set of points satisfies the triangular inequality. They found that in two dimensions, $(2\cos(\pi/6)-\epsilon)$ or $(1.732-\epsilon)$ approximation problem is NP-hard for all $\epsilon > 0$. Further, they proved that their algorithm is best possible with respect to the approximation bound, if $P \neq NP$.

2.4.2.2 Approximation Algorithm Based on Subgraph Technique for NP-complete Problems

Hochbaum *et al.* [63] carried out this study on selected approximation algorithms to wide variety of NP-complete problems. In their research, the basic approach is to identify a subgraph out of a complete graph satisfying certain constraints such that the length of the longest edge included in the subgraph is minimized. Moreover, their approximate algorithms for *k*-clustering and *k*-center delivered approximate solutions guaranteed to remains present within a constant factor 2 of the optimal solution. Their pragmatic approach is highly applicable in routing, location design and communication network design.

2.4.2.3 Approximation Algorithm for Pairwise Clustering and Central Clustering

Feder *et al*. [64] defined two measures of cluster size. First measure is the maximum distance between pairs of points in the same cluster which is also recognize as pairwise cluster and the second measure is the maximum distance between points in each cluster and a chosen cluster center also recognize as the central cluster. They proved that approximation clustering in $d \geq 2$ dimensions is NP hard for $\alpha < 1.82$(for pairwise cluster) and $\alpha < 1.97$ (for central cluster). They also elucidated a 2-approximation algorithm of running time $O(n\log k)$, emanated from box decomposition method [65].

2.4.2.4 Approximation Algorithm Based on Geometric Technique

Capoyleas *et al.* [50] investigated minimizing the diameter of the cluster or minimizing the radii of the cluster problem based on the geometric technique. Just like to above stated referred problem, minimizing the maximum radius problem is also well known as *k*-center problem [57]. They study the minimizing the diameter or minimizing radii of the clusters problem in polynomial time $O(n^{6k})$, for any fixed value of *k* and for any monotonic increasing function of cluster diameter or radius. Further, they concluded that any two clusters in an optimal solution can be separated by a line. In a similar work align with this study Megiddo *et al.* [66] also proved that minimizing the maximum cluster area problem and minimizing the sum of all cluster areas problem is NP-complete problem.

2.4.2.5 Approximation Algorithm for Minimize the Maximum Diameter and Minimize the Maximum Radius

There are a number of researchers addressed the clustering problem where the core objective is to minimize the maximum diameter or minimize the maximum radius of a cluster. Plesnik [67] disclosed that minimizing the maximum diameter and minimizing the total diameter, even for k=2, cannot be efficiently approximated to within factors less than 3/2 and 5/4 respectively unless P = NP.

Megiddo [68] presented an $O(n \log^3 n)$ algorithm for the continuous *p*-center problem and an $O(n \log^2 n \log\log n)$ algorithm for a weighted discrete *p*-center problem. Charikar *et al.* [69] modelled deterministic and randomized incremental version of the clustering problem for minimizing the maximum radius. Agarwal *et al.* [70] identified an $(1+\epsilon)$ approximation algorithm for the *k*-center problem, with running time complexity of $O(nlogk) + (k/\epsilon)^{O(k^{1-1/d})}$. Agarwal *et al.* [71] studied projective clustering problems with objective a subtle to cover the set *S* (set of *n* point in $\mathbb{R}^d$) by *k* hyper-strips (hyper-cylinders) in a manner such that maximum width of a hyperstrip (maximum diameter of a hyper-cylinder) is minimized, for positive integer value of *k*. Bădoiu *et al.* [72] investigated an $(1+\epsilon)$ approximation algorithm for the *k*-center clustering and *k*-median clustering problems in Euclidean

space based on the construct of core-sets. Running time of the algorithm has linear or near linear dependency on the number of points and the dimension, and exponential dependency on $1/\epsilon$ and *k*.

2.4.3 Approximation Algorithms for MSDC and MSRC problem

In this subsection, approximation algorithms for MSDC and MSRC problems are discussed.

2.4.3.1 Logarithmic Approximation Algorithm to Minimize the Sum of Diameter (2000)

Doddi *et al.* [29] discovered a logarithmic approximation algorithm for minimizing the sum of diameter problem. They regarded that problem of partitioning the nodes of a complete edge weighted graph into *k* clusters is to minimize the sum of diameters of the clusters. Algorithm 2.5 is based on the novel concept that merging pairs of intersecting clusters do not increase the total diameter of clusters, assuming clusters are pairwise disjoints.

Lemma 1. *Let I be an instance of minimum sum of diameter clustering given by the edge weighted complete graph G(V, E) and integer k. Let C = {C_1, C_2,. . . . C_k} be a collection of subsets of V such that their union is V and the sum of the diameters of all the subsets in C is ψ. Suppose C_i and C_j ($i \neq j$) are two sets in C such that $C_i \cap C_j \neq \phi$. Then total diameter of the collection C obtained* by deleting C_i *and* C_j from *C′ and adding the set $C_i \cup C_j$ is at most ψ.*

Algorithm 2.5 Approx*CMSD*($\boldsymbol{G(V,E)}, \boldsymbol{k}$)

Input: Complete edge weighted graph *G*, number of clusters *k*
Output: *k* clusters with minimum sum of diameter

1. Initialize *i* = set of |*V*| singleton clusters. $D = \{\{v\}: \ v \in V\}$
2. **while** (|*D*| > 10*k*) **do**
3. Assign *N*= set of vertices obtained by selecting one arbitrary vertex from each set of *D*
4. *C* = FindCover(*G*(N,*E*),*k*)
5. *D* = Merge(*D*, *C*)
6. **end while**
7. **return**(*D*)

Subroutine (1): *FindCover* $(G(N,E),k)$

This subroutine return a set of no more than $3k[1+(\ln(|N|/k))]$ clusters which cover *N* with cost no more than $3k[1+(\ln(|N|/k))](1+\epsilon)OPT$. This subroutine call ParametricBMCP $(G(N,E),k)$

Subroutine(2): *ParametricBMCP*$(G(N,E),k)$

The Budget Maximum Cover Problem (BMCP) selects a subcollection of sets so that the total cost of the selected sets is at most *B* and the number of elements covered by the chosen sets is a maximum. This subroutine generates a set of a set of no more than 3*k* clusters covering (1-1/*e*)|*N*| or more vertices from *N* with cost no more than $3(1+\epsilon).OPT$ for any fixed $\epsilon > 0$. This subroutine call TransformToSetCover $(G(N,E),k,f)$

Subroutine(3):*TransformToSetCover*$(G(N,E),k,f)$

This subroutine accepts an instance of CMSD along with a nonnegative value *f*, and produces an instance of the weighted set cover problem. With base set *Q*, and collection *W* of nonempty subsets of *Q*, each with a weight. This transformation is executed in polynomial time and it's based on the fact that OPT(*I*′)) < 2 OPT(*I*) + *f*, where OPT(*I*) and OPT(*I*′) denotes the optimal solution value of CMSD problem and weighted set cover problem respectively.

Subroutine (4):*Merge*(*D*,*C*)

This subroutine uses Lemma 1 and merges element of set *C* and element of Set *D* such that the overall cost is no more than the sum of the costs of set *C* and set *D*. They presented a logarithmic approximation algorithm that has no more than 10*k* clusters such the total diameter of generated clusters is within a factor *O*(log (*n*/*k*)) of the optimal value for *k* clusters, where *n* is the number of nodes in the complete graph. Their results rely on transformation of the problem into an instance of the weighted set cover problem and budget maximum coverage problem [73]. They identified an approximation algorithm for *k* clusters, such that cost of total diameter is at most twice the optimal value (for fixed value of *k*). They also addressed the NP-hardness of minimum sum of diameter

clustering problem (satisfying the triangle inequality) and proved that it is NP-hard to approximate the cost of total diameter to within a factor $2-\epsilon$ of the optimal value (for $\epsilon > 0$). Finally, they presented an polynomial time algorithm considering the underlying graph as a tree with edge weights, algorithm runs in $O(k^2n^3)$ and uses $O(kn^2)$ space.

2.4.3.2 Approximation Algorithm for Metric Version of MSDC & MSRC Problem

Charikar *et al.* [22] presented a primal–dual based constant factor approximation algorithm for the points in a metric space to minimize the sum of cluster diameters or the sum of cluster radii. They extended *k*-median approximation algorithms of Jain *et al.* [74] and Charikar *et al.* [75]. Algorithm 2.6 is a greedy algorithm that achieves a logarithmic approximation and applicable if distance function is asymmetric. They show that there exists a polynomial time randomized algorithm that achieves a (3.504 + ϵ) approximation for the sum-radii problem using at most *k* clusters, in running time of $n^{O(1/\varepsilon)}$.

They also proved that $\propto$ approximation algorithm minimizing the sum of radii clustering problem yields an $2 \propto$ approximation to minimize the sum of diameter clustering problem.

Algorithm 2.6 ApxAlgorithm(*N*)

---

1. Guess the largest *l* clusters in the optimal solution. $k' = k - l$. The remaining steps find at most k' clusters to cover the points outside the guessed *l* clusters.
2. Run algorithm primal-dual fixed cost sum-radii with all fixed costs set to *z*, perform a binary search on *z* to identify two values z_1 and z_2; such that the algorithm produces $\leq k'$ clusters for z_1 and $\geq k'$ clusters for z_2.
3. Let T_1 and T_2 be the set of tight clusters obtained by the algorithm for z_1 and z_2 and Let $\bar{F}_1 \subseteq T_1$ and $\bar{F}_2 \subseteq T_2$ be the set of original clusters picked in the solution for z_1 and z_2. Identify clusters in $\bar{F}_2$ that are disjoint from all clusters in $\bar{F}_1$. Add these clusters to one at a time to $\bar{F}_1$, until $|\bar{F}_2| = k'$.
4. Let F_1 denote the final value of $\bar{F}_1$ and $F_2 = \bar{F}_2$. Let S_1 and S_2 represent the solutions respectively with $|F_1| = k_1 \leq k'$ and $|F_2| = k_2 \geq k'$. Expressing k' as combination of $k_1$and k_2 with some coefficients *a* and *b*.
5. $a + b = 1, a.k_1 + b.k_2 = k'$

6. $a = \frac{k_2 - k'}{k_2 - k_1}$, $b = \frac{k' - k_1}{k_2 - k_1}$
7. Group the clusters, each group containing one cluster from F_1 and one or more cluster from F_2.
8. Apply probability distribution on the set of numbers n_q and allocation of clusters is determined by the disjoint sets S_1, S_2.
9. Return the better of the two solutions S_1 and the solution produced as the output of the previous step.

LP Formulation for the sum-radii problem: For every center *i* and radius *r*, $y_i^{(r)}$ is an indicator variable that indicates if there is a cluster of radius *r* centered at *i*.

$$\min \sum_i \sum_r r.y_i^{(r)} \tag{16}$$

$$\forall \text{ j} \sum_i \sum_{r:d(i,j)\leq r} y_i^{(r)} \geq 1 \tag{17}$$

$$\sum_i \sum_r y_i^{(r)} \leq k \tag{18}$$

Dual of the LP Problem is

$$\max \sum_j \propto_j - k.z \tag{19}$$

$$\forall\, i, r \sum_{j:d(i,j)\leq r} \propto_j \leq r + z \tag{20}$$

LP Formulation for the fixed cost sum-radii problem (f_i is fixed cost):

$$\min \sum_i \sum_r y_i^{(r)}.(r + f_i) \tag{21}$$

$$\forall \text{ j} \sum_i \sum_{r:d(i,j)\leq r} y_i^{(r)} \geq 1 \tag{22}$$

Dual of the fixed cost sum-radii LP Problem:

$$\max \sum_{j} \propto_j \tag{23}$$

$$\forall\, i, r \sum_{j:d(i,j)\leq r} \propto_j \leq r + f_i \tag{24}$$

LP formulations of the sum-radii problem can be transformed similar to the LP formulations of the fixed costs sum-radii problem by setting the fixed costs f_i = *z*. The constraints in the fixed costs sum-radii dual LP equation (23)–(24) are exactly the same as the constraints in the sum-radii dual LP equation (19)–(20). Thus, a feasible solution to the fixed costs sum-radii problem is a feasible solution to the sum-radii problem.

2.4.3.3 Approximation Algorithm for Geometric Disk Covering Problem

Research work of Tov *et al.* [30] is concerned with geometric disk covering problem [76]. Their research work is based one of the geometric disc covering problem, placement of base stations in wireless network design. In base station placement problem, set of clients are to be covered by collection of disks of variable radii around a base station location, such that sum of radii of disk is minimized. Similar problem like fixed radius covering problem with given potential server locations is considered by Glasser *et al.* [77].

Disk covering Problem: Let us consider $X = \{x_1, x_2, \ldots x_m\}$ representing base locations for placing base stations (servers) and $Y = \{y_1, y_2, \ldots y_m\}$ representing the clients. A base station located at x_i has a certain transmission range R_i. A client node y_j is covered by a base station placed at x_i, if y_j is contained in the disk of radius R_i centered at x_i or falls on its boundary. The disk covering problem is to finding a collection of servers covering all the clients. Where as in minimum sum of radii cover problem, objective is to select the transmission radii R_i such that all the clients are covered and the sum of the transmission ranges is minimized. They presented a polynomial time algorithm of time complexity $O((n+m)^3)$ based on dynamic programming for the 1-dimensional minimum sum of radii cover problem. Lastly, they also identified polynomial time approximation scheme for the minimum sum of radii cover problem with approximation ratio of $(1+6/k)$ and time complexity

of $O(k^2(nm)^{\gamma+2})$. Polynomial time approximation scheme is based on a modified variant of the hierarchical grid-shifting technique [78].

2.4.3.4 Approximation Algorithms for Geometric Version of min-size *k*-clustering Problem

Bilo *et al.* [75] studied geometric versions of the min-size *k*-clustering problems which generalize clustering to minimize the sum of cluster radii problem. They also studied PTAS for each instance (X, F, d, α) of the minimum sum of radii cover problem, where X is a set of n points with rational coordinates on the d-dimensional Euclidean space, F is a non-negative cost function associated with each point, and a constant value α. Their algorithm extended the idea of plane subdivision from an algorithm of from Erlebach *et al.* [78] that approximates the minimum vertex cover of disk graphs. Their algorithm computes an $(1+\epsilon)$-approximate solution in $n^{(\alpha/\epsilon)^{O(d)}}$time. When the points to be clustered are located on a line, clustering to minimize the sum of cluster size problem can be solved in polynomial time.

2.4.3.5 Approximation Algorithms for Euclidean Version of min-cost *k*-cover Clustering Problem

Gibson *et al.* [80] showed that the Euclidean min-cost *k*-cover problem is solvable exactly in polynomial time (on the assumption of some model of computation). Optimal *k*-cover problem can be efficiently computed using dynamic programming, as in [81–82]. Euclidean version of the problem is well examined in [29, 30] and it is extended to metric version of probabilistic partitions [83–84]. Other researchers also worked in the same direction, but their research work is associated with some limitations like, Alt *et al.* [85] consider a class of geometric facility location problems and showed that the NP-hardness result for the MSRC problem can be extended to any $\alpha > 1$. Alt *et al.* [85] presented $O(n^4 \log n)$ fast constant-factor approximation algorithms that cost (4^{α} OPT) for the MSRC problem and also consider various related problems. Wei-lin *et al.* [86] gave a polynomial-time approximation algorithm for the ω-constrained *k*-facility location problem with approximation ratio

ω+1+ϵ and proved that ω-constrained facility location problem cannot be approximated with in $1 + \ln\sqrt{\omega - 1}$.

In the geometric version of min-cost *k*-cover problem, the optimal solution is a set of *k* disks, each of which is centered on some input point and each of which has a radius that is the distance between two input points. If input points have integer coordinates, cost of such a solution is the sum of square roots of integers. Two sums of square roots of integers can be compared in polynomial time is an open problem [87–88]. Comparing two sums of square root of integer in polynomial time is an open problem [80].

Theorem 7. [**80**] There is a polynomial time algorithm that, given a set *P* of points in the plane and an integer $k \geq 1$, returns an optimal *k*-cover of *P*.

They showed that the Euclidean min-cost *k*-cover problem ($\alpha = 1$) is solvable exactly in polynomial time (Theorem 7). Aspect ratio (Δ) of the input point set *P*, is the ratio of the maximum to minimum inter-point distance within *P*. When Δ is bounded by a polynomial in *n*, it yields a randomized algorithm that runs in $n^{O(\log n \cdot \log \Delta)}$ time returns an optimal *k*-cover of *P* with high probability. When Euclidean metric does not hold the model of computation, exact algorithm for the Euclidean metric can be translated into an approximation algorithm. Algorithm 2.7 runs in polynomial time in the input size and $\log(1/\epsilon)$ and returns a solution of cost at most $(1+\epsilon)$ times the optimal solution ($0 < \epsilon < 1$).

Lemma 2. [80] *Consider an optimal κ-cover D for some set $Q \subseteq P$ of points contained in a rectangle R. The rectangle R has a separator that intersects at most 12 disks in D.*

Algorithm 2.7: DC(*R*, *κ*, *ϕ*)

Input: A balanced rectangle *R*, an integer $\kappa \geq 0$, and a subset $T \subseteq D$.
Output: optimal *k*-cover

1. Create a Table ($P \cap R$, κ, *T*) if not created. $D' = \{ I \}$
2. **for** all choices of separator $l \in L(R')$ **do**

3. **for** all choices a set $D_0 \subseteq D$ of at most 12 disks (Lemma 2) that intersect l **do**
4. **for** all choices of $\kappa_1, \kappa_2 \geq 0$ such that $\kappa_1 + \kappa_2 + D_0 < \kappa$ **do**
5. Let R_1 and R_2 be two rectangles into which l partitions R'.
   Let $T_1 = \{D \in T \cup D_0 \mid D \text{ intersects } R_1\}$.
   Let $T_2 = \{D \in T \cup D_0 \mid D \text{ intersects } R_2\}$
6. **if** $|T_1| \leq \beta$ and $|T_2| \leq \beta$ **then**
7. Recursively call DC (R_1, κ_1, T_1) U DC (R_2, κ_2, T_2)
8. **if** cost (D_0 U Table R_1, κ_1, T_1) U Table (R_2, κ_2, T_2)) < cost (D')
then update $D' \leftarrow D_0 \cup \text{Table } (P \cap R_1, \kappa_1, T_1) \cup \text{Table } (P \cap R_2, \kappa_2, T_2)$
9. Assign Table (R, κ, T) by D'
10. **return**.

Description of the algorithm:

The algorithm begins with a rectangle containing all the points and cuts it into two smaller rectangles by selecting a separator line and solves the sub-problems corresponding to smaller rectangles recursively, assuming disc I is of infinite radius. Consider an instance of the Euclidean minimum sum of radii covering problem to compute an optimal k-cover of P of n points in the plane. Let D be the set of discs, whose center is some $p \in P$ and radius is $|pq|$ for $q \in P$, then $|D| = n^2$. In a similar manner, if D is the set of distinct maximum cluster then $D \leq n^2$.

Algorithm 2.7 is a dynamic programming algorithm which employs balanced rectangles to define the subproblems. A rectangle(R) is said to be balanced if its width is at least a third of its length, i.e. length(R) $\geq$ 3 * width(R). A vertical (horizontal) line is critical if it passes through a point in P or a point of vertical (horizontal) tangency of some disk in D. A separator for a (balanced) rectangle is any line which is perpendicular to its longer side and cuts it in the middle third of its longer side. Procedure compress(R) accepts a balanced rectangle R' that contains at least two of the points in P and returns a balanced rectangle R such that (a) R' is contained in R, (b) R' contains $P \cap R$, and (c) for any separator for R', there are points of $P \cap R$ in both of the open half-spaces that it bounds (and consequently, any separator for R' partitions $P \cap R$ into two nonempty subsets).

2.4.3.6 Approximation Algorithms for Metric Version of min-cost *k*-cover Clustering Problem

Gibson *at el.* [89] explored the metric versions of clustering to minimize the sum of radii problem. They generalized the algorithmic approach of Gibson *et al.* [30] to the metric case and then probabilistically partitioned the metric into sets. For the *k*-cover metric problem, they obtained an exact algorithm of running time is $n^{O(\log n \cdot \log \Delta)}$, where Δ (aspect ratio is the ratio between maximum interpoint distance and minimum interpoint distance. Proposed algorithm is randomized in nature and succeeds with high probability. When, Δ is bounded by a polynomial in *n*, the running time of the algorithm is quasi-polynomial. In their research work they proposed an randomized algorithm of $n^{O(\log n \cdot \log \Delta)}$ time for *k*-cover problem, that takes input a set *P* of *n* points in a metric space, an integer *k*, and a parameter ϵ. Cost of the proposed algorithm with probability at least ½ is within a multiplicative factor of $(1+\epsilon)$ of the optimal *k*-cover.

2.4.3.7 Approximation Algorithms for Euclidean Version of Minimum Sum of Radii Cover Problem

Gibson *et al.* [90] showed that the Euclidean min-cost *k* cover problem is solvable exactly in polynomial time, under the assumption that the cost of any two candidate covers can be compared in polynomial time. Proietti *et al*. [91] also considered a problem closely related to the metric min cost *k*-cover problem or *k*-radius problem, they proved that the problem is NP-hard.

They showed that there is an algorithm that, given a set *P* of *n* points in the plane and an integer $k \geq 1$, runs in $O(n^{881}.T(n))$ time and returns an optimal *k*-cover of *P*. *Here*, $T(n) \geq 1$ is an upper bound on the time needed to compare the costs of two subsets of *D*, each of size at most *n*, and *D* is the set of n_2 disks whose center is some $p \in P$ and whose radius is $|pq|$ for some $q \in P$. They showed that there is an approximation algorithm if the model of computation is bypassed. Given a set *P* of points in the plane, an integer $k \geq 1$, and a parameter $0 < \epsilon < 1$, their exists an algorithm that runs in time polynomial in the input size and $\log(1/\epsilon)$ and returns a *k*-cover of *P* whose cost is at most

$(1+\epsilon)$ times the cost of an optimal *k*-cover. They presented an algorithm for minimum sum of radii cover problem that runs in time $n^{O(1)}$.T(*n*) and returns an optimal solution. Here, *C* is set of clients and a set *F* of facilities in $\mathbb{R}^2$, $n = |F \cup C|$, and $T(n) \geq 1$ is an upper bound on the time required to compare any two subsets of *D*; *D* consists of the set of $O(n^2)$ disks with center at some $p \in F$ and radius $|pq|$ for some $q \in C$ and there is no upper bound on the number of disks.

2.4.3.8 Approximation Algorithms for Metric Version of Clustering to Minimize the Sum of Radii Clustering

Behsaz *et al.* [92] explored metric versions of clustering to minimize the sum of radii. They presented a polynomial time exact algorithm for minimum sum of radii cover problem that runs in $O(\log^2 n)$ for metrics of unweighted graphs, assuming no singleton clusters are allowed. They searched out an $(1+\epsilon)$ PTAS for the MSDC problem that runs in time $n^{O(1/\epsilon)}$, set *V* contains *n* points in $\mathbb{R}^2$, an integer *k*, and an error bound $\epsilon > 0$. For the fixed value of *k*, they presented an exact algorithm for MSDC problem of $O\left(k^2 n^{k^2+k+2}\right)$.

2.5 Conclusion

This chapter presents a detailed survey on exact and approximate algorithms for minimum sum of diameter and minimum sum of radii clustering algorithms. This chapter covers various algorithm which are scattered in the various literatures. This chapter investigates the algorithm on the following key factors: formulation/technique, year of evolution, clustering criteria, approach and time complexity. In this research, we presented a comparative overview of the algorithm in tabular form that identifies issues, constraints, assumptions and challenges in the algorithm. MSDC and MSRC clustering algorithm is still an open problem due to the existence of many inherent constraints and limitation of existing algorithms.

Table 2.2: Approximation algorithms for minimum sum of diameter clustering and minimum sum of radii clustering: A Comparative Overview

Year	Author Name	Clustering Algorithm	Approach/Concept	Constraint/Assumption/ Issues	(TC)/ (AF)
2000	Doddi *et al*. [29]	Logarithmic approximation algorithm for MSDC Approximation algorithm for MSDC Polynomial time algorithm for MSDC	Parametric budget maximum coverage Minimum cost set cover approach Dynamic programming	For no more than 10k clusters For fixed value of k. Underlying graph is assume as a tree with edge weights	AF: $O(\log(n/k))$ AF: 2 TC:$O(k^2n^3)$
2004	Charikar *et al*. [22]	Approximation algorithm for the metric version of MSRC	Based on primal dual algorithm	For at most k clusters	AF: $(3.504+\epsilon)$ TC: $n^{O(1/\epsilon)}$
2005	Tov *et al*. [30]	Polynomial time algorithm for 1-dimensional minimum sum of radii cover problem Polynomial time approximation scheme for minimum sum of radii cover problem	Dynamic program ming Geometric disk covering and hierarchical grid-shifting technique	Clients and servers are located on a straight line constant γ is dependent on k	TC: $O((n+m)^3)$ AF: $(1+6/k)$ TC: $O(k^2.(nm)^{\gamma+2})$
2005	Bilo *et al*. [79]	Polynomial time algorithm for min-size k-clustering problem Polynomial time approximation scheme for each instance (X, F, d, α) of the geometric min-size k-clustering	Expressing the problem as integer linear programming Plane subdivision technique with approximating the minimum vertex cover of disk graph	Points should be located on a line For points in Euclidean space of constant dimension	TC: $n^{O(\lambda^4+\xi)}$ AF: $(1+\epsilon)$ TC: $n^{(\alpha/\epsilon)^{O(d)}}$
2008	Gibson *et al*. [80]	Approximation algorithm for minimum cost k-cover problem for Euclidean metric	Dynamic programming algorithm which uses balanced rectangle as a sub problems.	Model of computation is not hold	AF: $(1+\epsilon)$ TC: polynomial in input size and $\log(1/\epsilon)$

2010	Gibson *et al*. [89]	Exact algorithm for minimize cost *k*-cover problem for metric case Approximation algorithms for minimize cost *k*-cover problem for the metric space	Probabilistic partitions Discretization of problem into several instances of exact metric *k*-cover problem	Assuming Δ is polynomial bounded by *n* Model of computation does not hold	TC: $n^{(logn.log\Delta)}$ AF: $(1+\epsilon)$ TC: $n^{(logn.log\frac{n}{\epsilon})}$
2012	Gibson *et al*. [90]	Polynomial time exact algorithm for Euclidean min-cost *k*-cover problem Approximation algorithm for the Euclidean min-cost *k*-cover problem Polynomial time algorithm for Euclidean minimum sum of radii cover problem	Structure possessed by optimal solutions is eminently separable by a line and dynamic programming Introducing the proxy cost for each disk and comparing the proxy cost of disk rather than actual cost Recursive procedure, disk are centered only at the facilities and not the clients	Assuming the cost of two subsets of *D* can be compared polynomial time Model of computation does not hold No upper bound on number of disk and a point can be client or facility at a time	TC: $O(n^{881}$ .T $(n))$ AF: $(1+\epsilon)$ TC: Polynomial in the input size and log $(1/\epsilon)$ TC: $n^{O(1)}$ .T (n)
2012	Bahaz *et al*. [92]	Polynomial time exact algorithm for minimum sum of radii clustering problem for metrics of unweighted graph Polynomial time approximation scheme algorithm for MSDC problem for $\mathbb{R}^2$ Polynomial time exact algorithm for the MSDC problem	Reducing the minimum sum of radii problem polynomial time to the minimum sum of radii problem for connected graphs In optimal solution convex hulls of the clusters are disjoint Distance between two vertices of different cluster is greater than sum of diameters then vertices can be separated	Graph should be polynomially bounded, no singleton cluster is allowed, graph should be connected Clusters are not necessary defines by a disc Applicable only for constant value of *k*	TC: $n^{o(log^2 n)}$ AF: $(1+\epsilon)$ TC: $n^{O(1/\epsilon)}$ TC: $O(k^2 n^{k^2+k+2})$

Chapter 3

Reduction of Clustering Problem as SAT Statement

3.1 Introduction

3-Clustering is partitioning a set of entities into three non-empty clusters such that there sum of diameter is minimum. Similarly, k-clustering is partitioning a set of entities into k non-empty clusters such that there sum of diameter is minimum. 3-clustering and k-clustering problem are NP-complete problem. k-clustering problem is applicable in many real-life applications. k-clustering plays important role in social networking. In this chapter, we introduce the concept of reduction of 3-clustering problem to 3-SAT formulation and k-clustering problem to k-SAT formulation.

3.2 Reduction of 3-clustering Problem to 3-SAT Formulation

Let O = $\{O_1, O_2, . , . , O_N\}$ denote a set of N = |O| entities and $D = \{ d_{kl} / t \leq k \leq N, 1 \leq 1 \leq N \}$ a set of dissimilarities between pairs of these entities. A dissimilarity d_{kl} is a real number and satisfies to the conditions $d_{kt} \geq 0$, $d_{kk} = 0$, and $d_{kl} = d_{lk}$ for k , l = l , 2 , . . . , N .A partition $P_M = \{C_1, C_2, C_3 \}$ of the entities of O into 3 clusters is such that no cluster is empty, any pair of clusters has an empty intersection and the union of all clusters is equal to O. In this chapter, three clustering problem for minimum sum of diameter is reduced into 3-SAT statements using belonging approach.

3.2.1 Belonging Approach

3-Clustering is partitioning a set of $O = \{O_1, O_2, \ldots, O_n\}$ entities into 3 clusters C_1, C_2 and C_3 such that no cluster remains empty, any pair of clusters has an empty intersection and the union of all clusters is equal to O. A belonging formulation [94], reduces 3-clustering problem to 3-SAT formulation. Consider the three clusters C_1, C_2 and C_3, and r_1, r_2, r_3 are radius respectively such that $r_1 > r_2 > r_3$. An association of variable x_i for each entity is formulated as follows:

$$x_i = \begin{cases} p_i, & if\ O_i \in C_1 \\ q_i, & if\ O_i \in C_2 \\ r_i, & if\ O_i \in C_2 \end{cases} \tag{25}$$

Thus, if entity O_j belong to cluster C_1 then $p_j = 1,\ q_j = 0$ and $r_j = 0$

3.2.2 Formulation of 3-clustering problem as 3-SAT Formulation

Constraint type 1: If $d_{kl} > r_1$, then O_k and O_l together cannot belong to the same cluster C_1, C_2 and C_3, generated constraints are as:

a. $\text{if}(O_k \in C_1)\ \text{then}\big((O_l \in C_2)\text{or}(O_l \in C_3)\big) \Rightarrow (p_k \wedge (q_l \vee r_l) = 1)$

b. $\text{if}(O_k \in C_2)\ \text{then}\big((O_l \in C_1)\text{or}(O_l \in C_3)\big) \Rightarrow (q_k \wedge (p_l \vee r_l) = 1)$

c. $\text{if}(O_k \in C_3)\ \text{then}\big((O_l \in C_1)\text{or}(O_l \in C_2)\big) \Rightarrow (r_k \wedge (p_l \vee q_l) = 1)$

d. $\text{if}(O_l \in C_1)\ \text{then}\big((O_k \in C_2)\text{or}(O_k \in C_3)\big) \Rightarrow (p_l \wedge (q_k \vee r_k) = 1)$

e. $\text{if}(O_l \in C_2)\ \text{then}\big((O_k \in C_1)\text{or}(O_k \in C_3)\big) \Rightarrow (q_l \wedge (p_k \vee r_k) = 1)$

f. $\text{if}(O_l \in C_3)\ \text{then}\big((O_k \in C_1)\text{or}(O_k \in C_2)\big) \Rightarrow (r_l \wedge (p_k \vee q_k) = 1)$

Constraint type 2: If $r_1 > d_{kl} > r_2$, then O_k and O_l together cannot belong to the same cluster C_2 and C_3, generated constraints are as:

a. $\text{if}(O_k \in C_2)\ \text{then}\big((O_l \in C_1)\text{or}(O_l \in C_3)\big) \Rightarrow (q_k \wedge (p_l \vee r_l) = 1)$

b. $\text{if}(O_k \in C_3)\ \text{then}\big((O_l \in C_1)\text{or}(O_l \in C_2)\big) \Rightarrow (r_k \wedge (p_l \vee q_l) = 1)$

c. $\text{if}(O_l \in C_2)\ \text{then}\big((O_k \in C_1)\text{or}(O_k \in C_3)\big) \Rightarrow (q_l \wedge (p_k \vee r_k) = 1)$

d. $\text{if}(O_l \in C_3)\ \text{then}\big((O_k \in C_1)\text{or}(O_k \in C_2)\big) \Rightarrow (r_l \wedge (p_k \vee q_k) = 1)$

e. $(p_k \vee p_l) = 1$

Constraint type 3: If $r_2 > d_{kl} > r_3$, then O_k and O_l together cannot belong to the cluster C_3, generated constraints are as:

a. $\text{if}(O_k \in C_3)\ \text{then}\big((O_l \in C_1)\text{or}(O_l \in C_2)\big) \Rightarrow (r_k \wedge (p_l \vee q_l) = 1)$

b. $\text{if}(O_l \in C_3)\ \text{then}\big((O_k \in C_1)\text{or}(O_k \in C_2)\big) \Rightarrow (r_l \wedge (p_k \vee q_k) = 1)$

c. $(p_k \vee p_l) = 1$

d. $(q_k \vee q_l) = 1$

Constraint type 4: If $r_3 > d_{kl}$ then there is no restriction, O_k and O_l can belong to any cluster.

3.2.3 Complexity Computation

The present research assumed a complete graph $G(V, E)$, having n vertices. Number of edges in the graph G will be $n*(n-1)/2$ or will order of $O(n^2)$. Every edge belongs to Constraint type 1/ type 2/ type 3/ type 4 and generated constraints are in the form of 3-SAT. Hence, number of constraints/clauses generated will be 6, 5, 4, 0 respectively. So, for graph G, number of constraints/clauses generated will order as $O(6n^2)$. Number of nodes (vertices) n can belong to 3 clusters so; numbers of generated variables are $3n$. In case of minimum sum of radii problem for 3 clusters, all subsets of distinct maximal clusters are generated in order of $O(n^6)$. Therefore, time complexity to reduce 3-cluster to 3-SAT for minimum sum of radii problem is $O(n^8)$.

3.3 Reduction of *k*-clustering Problem to *k*-SAT Statement

Social networks are social communities of the web, connected via electronic mail, websites and web logs, and networking applications such as Twitter, Facebook, or LinkedIn. Social network analysis maps and measures formal and informal relationships to understand what facilitate or impede the knowledge flows that bind interacting units. In social networks [23], "nodes" of the network are people and the "links" are the relationships between people. Nodes are also used to represent events, ideas, objects, or other things. Social network analysis practitioners collect network data, analyses the data and often produce maps or pictures that display the patterns of connections between the nodes of the network. These maps reveal characteristics of the network that guide

participants as they evaluate their network and plan ways to improve their collective ability to identify and achieve shared goals. Constraints provide guidance about the desired partition and make it possible for clustering algorithms to increase their performance.

3.3.1 *k*-clustering Problem Statement

Let $O = \{O_1, O_2, \ldots, O_N\}$ denote a set of $N = |O|$ entities and $D = \{ d_{ij} / i \leq k \leq N, 1 \leq j \leq N \}$ a set of dissimilarities between pairs of these entities. A dissimilarity d_{ij} is a real number and satisfies to the conditions $d_{ij} \geq 0$, $d_{ii} = 0$, and $d_{ij} = d_{ji}$ for $i, j = 1, 2, \ldots, N$. A partition $P_M = \{C_1, C_2, \ldots, C_k\}$ of the entities of O into K clusters is such that no cluster is empty, any pair of clusters has an empty intersection and the union of all clusters is equal to O.

3.3.2 Transformation of Social Network Concepts into SAT Statement

Bonding and bridging are two different important connectivity and measures in social network. In Figure 3.1 Bonding denotes connections in a tightly bind group. Bridging denotes connections to another cluster. Social network analysis literature, bonding and bridging are often called closure and brokerage respectively. Analyzing network data to measure bonding and bridging helps to predict important outcomes such as efficiency and innovation: bonding indicates a sense of trusted community where interactions are familiar and efficient; bridging indicates access to new pattern or group.

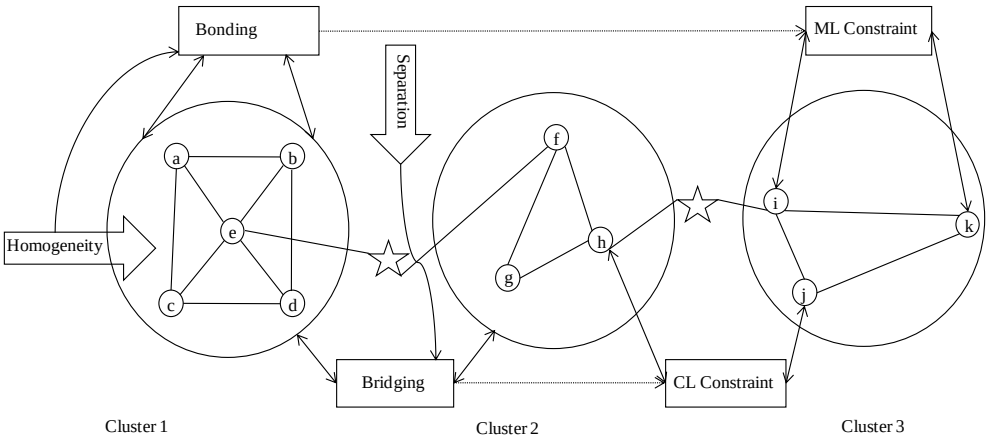

Figure 3.1: Mapping of bonding and bridging to ML & CL constraints in a social network

Concept of social networking can be transformed into mathematical model. The transformation process is drawn in Figure 3.2. Transformation of business logic on the basis of attributes of objects/actor to the properties like homogeneity and separations. These properties homogeneity and separations are transformed into bonding and bridging respectively to construct social network. Social network concepts are transformed into must link and can-not link constraints. These ML and CL constraints are represented in a mathematical form of as a SAT statement.

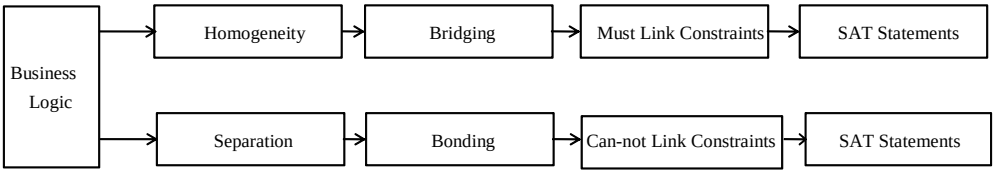

Figure 3.2: Transformation of business logic into SAT statements

3.3.3. Belonging Approach

For *k*-clustering, a belonging approach put forward *k*-clustering problem as *k*-SAT formulation. *k*-clustering is partitioning a set $O = \{O_1, O_2, \ldots, O_n\}$ entities into *k* clusters $C_1, C_2, \ldots, C_k$ cluster such that no cluster remains empty, any pair of clusters has an empty intersection and the union of all clusters is equal to *O*. Consider $C_1, C_2, \ldots, C_k$ clusters and $r_1, r_2, \ldots, r_k$ are cluster radius. Such that $r_1 > r_2 > \ldots > r_k$. An association of variable x_i for each entity is formulated as:

$$x_i = \begin{cases} p_i, & if\ O_i \in C_1 \\ q_i, & if\ O_i \in C_2 \\ r_i, & if\ O_i \in C_3 \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots. \\ t_i, & if\ O_i \in C_{k-1} \\ s_i, & if\ O_i \in C_k \end{cases} \tag{26}$$

Thus, if entity O_j belong to cluster C_1 then $p_j = 1,\ q_j = 0$ and $r_j = 0$

3.3.4 Formulation of *k*-clustering Problem as *k*-SAT Formulation

Constraint type 1: If $d_{kl} > r_1$, then O_k and O_l together cannot belong to the same cluster $C_1, C_2, \ldots, C_k$, in this case generated constraints are as:

a. if$(O_k \in C_1)$ then$\big((O_l \in C_2)$or$(O_l \in C_3)$or$\ldots$or$(O_l \in C_k)\big) \Rightarrow (p_k \wedge (q_l \vee r_l \vee \ldots \vee s_l) = 1)$

b. if$(O_k \in C_2)$ then$\big((O_l \in C_1)$or$(O_l \in C_3)$or$\ldots$or$(O_l \in C_k)\big) \Rightarrow (q_k \wedge (p_l \vee r_l \vee \ldots \vee s_l) = 1)$

……up to *k* times for entity O_k

k. if$(O_l \in C_k)$ then$\big((O_k \in C_1)$and$(O_k \in C_2)$or$\ldots$or$(O_l \in C_k)\big) \Rightarrow (r_l \wedge (p_k \vee q_k \vee \ldots \vee_l t_i) = 1)$

Similarly, constraints are generated up to *k* times for entity O_l also.

So, Number of *k*-SAT constraints = 2*k*.

Constraint type 2: If $r_1 > d_{kl} > r_2$, then O_k and O_l cannot belong to the same cluster $C_2, C_3, \ldots, C_k$, generated constraints are as:

a. if$(O_k \in C_2)$ then$\big((O_l \in C_1)$or$(O_l \in C_3)$or$\ldots$or$(O_l \in C_k)\big) \Rightarrow (q_k \wedge (p_l \vee r_l \vee \ldots \vee s_l) = 1)$

b. if$(O_k \in C_3)$ then$\big((O_l \in C_1)$or$(O_l \in C_2)$or$\ldots$or$(O_l \in C_k)\big) \Rightarrow (r_k \wedge (p_l \vee q_l \vee \ldots \vee s_l) = 1)$

In the above stated case, *k*-SAT constraints are generated for *k*-1 times for entity $O_k$and O_l,

$(p_k \vee p_l) = 1$

In the above stated case, 2-SAT constraints are generated only for 1 time,

So, Number of constraints = 2(*k*-1) +1

Constraint type 3: If $r_2 > d_{kl} > r_3$, then O_k and O_l together cannot belong to the same cluster $C_3, C_4, \ldots, C_k$, generated constraints are as:

Similarly, *k*-SAT constraints are generated for 2(*k*-2) times for entity $O_k$and O_l and 2-SAT constraints are generated 2 times

So, Number of constraints = 2(*k*-2) +2

Constraint type k: If $r_{k-1} > d_{kl} > r_k$, then entities O_k and O_l are not belong to the same cluster C_k.

Similarly, total Number of constraints = 2(k-(k-1)) +k-1

Constraint type k+1: If $r_k > d_{kl}$ then there is no restriction, O_k and O_l can belong to any cluster.

3.3.5 Complexity Computation

The present research assumed that a complete graph $G(V, E)$, having n vertices. Number of edges in the graph G will be n*(n-1)/2 or will order of $O(n^2)$. Every edge will belong to constraint type 1/ type 2/ …./ type k+1 and generated constraints are in the form of k-SAT. Hence, number of constraints generated is order of $O(k)$, so for n^2 edges order will be $O(kn^2)$. Number of nodes (vertices) n can belong to k clusters so, number of variables generated are 3n. In case of minimum sum of radii problem for k clusters, all subsets of distinct maximal clusters are generated in $O(n^{2k})$. Thus, complexity to reduce k-cluster problem to k-SAT problem for minimum sum of radii problem is $O(kn^{2k+2})$.

3.4 Conclusion

Social network analysis is fast-growing field data mining. Reducing the social networking problem into Sat formulation is a good way to analyze the and investigates the network data. Boolean approach techniques is used for grouping of data items into two clusters for minimum sum of diameter clustering. Hansen [4] applied Boolean approach to find out minimum sum of diameter for two clusters by translating them into 2-SAT statement. Boolean approach is not sufficient to represent the constraint for 3-clustering or k-clustering. In this research, we extended the concept of reduction of 2-clustering to 2-SAT. This chapter presents a new formulation for reduction of 3-clustering to 3-SAT and k-clustering to k-SAT in polynomial time.

Chapter 4

Partitioning and Constraint 3-clustering Algorithm

4.1 Introduction

Partitioning is a fundamental problem with applications in several fields of study. In computer science, the partition problem is an NP-complete problem and it is also NP-Hard to find good approximate solutions for this problem. Zuckerman [48] shows that all of the problems (21 Problems) listed by Karp [17] are NP-Complete and have a version that's hard to approximate. Partitioning Problem is one of the problems in the listing of Karp's 21 Problems. These versions are obtained from the original problems by adding essentially the same, simple constraint. These problems are absurdly hard to approximate. Karp [17] also shows that one cannot even approximate $log^{(k)}$ of the magnitude of these problems to within a constant factor, where $log^{(k)}$ denotes the iterated logarithm, unless NP is recognized by slightly super polynomial randomized machines. Application of Partitioning problem are very broad. Partitioning problem are applied in various applications [94-95] like: In circuit and in VLSI design, in parallel processing, in combinatorial optimization, in scheduling of jobs to processors so as to minimize some cost and in cluster Analysis

4.2 Partitioning Problem

A Partition of a set *U* is a subdivision of the set into subsets that are disjoint and exhaustive, i.e. every element of *U* must belong to one and only one of the subsets. The subsets A_i in the partition are called cells. Thus $\{A_1, A_2, \ldots\ldots, A_r\}$

is a partition of U if two conditions are satisfied: (1) $\{A_1 \cap A_j = \emptyset\}$ if $\{i \neq j\}$ (the cells are disjoint) and (2) $\{A_1 \cup A_2 \cup \ldots\ldots \cup A_r\}$ (the cells are exhaustive).

4.2.1 Terminology

Stirling numbers of the first kind: They commonly occur in combinatorics, where they appear in the study of permutations. Stirling numbers of the first kind [96, 97] are written with a small *s*. The Stirling Numbers of the First Kind can be defined as *s*(*n*, *k*) ways of partitioning a set of *n* elements into *k* disjoint cycles. Stirling numbers of the first kind counts the number of ways to arrange *n* objects into *k* cycles instead of subsets. Stirling numbers of the first kind are represented as

$$\left[\begin{matrix} n \\ k \end{matrix} \right] \text{ means “}n\text{ cycle's }k\text{”}$$

Recursive relation for Stirling numbers of the first kind is

$$\left[\begin{matrix} n+1 \\ k \end{matrix} \right] = n \left[\begin{matrix} n \\ k \end{matrix} \right] + \left[\begin{matrix} n \\ k-1 \end{matrix} \right] \tag{27}$$

Where $\left[\begin{matrix} n \\ k \end{matrix} \right] = |s(n,k)|$

For $k > 0$, with the initial conditions

$\left[\begin{matrix} 0 \\ 0 \end{matrix} \right] = 1$ and $\left[\begin{matrix} n \\ 0 \end{matrix} \right] = \left[\begin{matrix} 0 \\ n \end{matrix} \right] = 0$ for $n > 0$

Stirling number of the second kind: Stirling numbers of the second kind [7, 8] occur in the field of mathematics called Combinatorics and in the study of partitions. In mathematics, particularly in combinatorics, a Stirling number of the second kind is the number of ways to partition a set of n objects into *k* non-empty subsets and is denoted by *S*(*n*, *k*). Stirling numbers of the second kind are written with a Capital *S*. Recursive relation for Stirling numbers of the second kind is

$$\left\{ \begin{matrix} n+1 \\ k \end{matrix} \right\} = k \left\{ \begin{matrix} n \\ k \end{matrix} \right\} + \left\{ \begin{matrix} n \\ k-1 \end{matrix} \right\} \tag{28}$$

Where, $\left\{ \begin{matrix} n \\ k \end{matrix} \right\} = |S(n,k)|$

for $k > 0$, with the initial conditions

$\left\{ {n \atop k} \right\} = 1$When $n = k$ and $\left\{ {n \atop 1} \right\} = 1$ and for $n > 0$

Bell numbers: The Bell numbers describe the number of ways a set with n elements can be partitioned into disjoint, non-empty subsets. Starting with $B_0 = B_1 = 1$. Bell number satisfies the recurrence formula. Figure 4.1 show the number of partition generated for B_3.

$$B_{n+1} = \sum_{k=0}^{n} \binom{n}{k} B_k \tag{29}$$

Each Bell number is a sum of Stirling numbers of the second kind

$$B_{n+1} = \sum_{k=1}^{n} \left\{ {n \atop k} \right\} \tag{30}$$

where$\left\{ {n \atop k} \right\}$ is a Stirling numbers of the second kind $\left\{ {n \atop k} \right\} = |S(n,k)|$

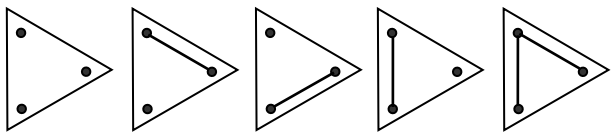

Figure 4.1: Number of partition generated for B_3

4.2.2 Bit wise Representation of Partition

In this method Partitions are represented in the form of bits. For the 3 partitioning of n-entities 3n bits are required. Each of the n bits represents one partition. When 3-partitioning is represented in bit form, a specific pattern is generated for 3-Partition for any value of n. In this pattern first n consecutive bits represent 1st Partition, next n consecutive bits represent 2nd Partition and last n consecutive bits represent 3rd Partition.

Rule of Belongingness: If an entity O_i belongs to Partition C_i then the i^{th} value of O_i is 1 otherwise it is zero.

Let us consider a set of 5 entities $U = \{a, b, c, d, e\}$,

Let us a take a sample

$P_1 = \{a, b, c\}$, $P_2 = \{e\}$ & $P_3 = \{d\}$

Above partition can be represented in bit format as

Partition 1					**Partition 2**					**Partition 3**				
a	*b*	*c*	*d*	*e*	*a*	*b*	*c*	*d*	*e*	*a*	*b*	*c*	*d*	*e*
1	1	1	0	0	0	0	0	0	1	0	0	0	1	0

Figure 4.2: Bit wise representation of a partition sample

4.2.3 Rule-Set for 3-BitPartition

1. Total number of 1's in 3n bits is always equal to n.(so rest 2n bits are always zero)
2. Among the three cluster, on the same bit position exactly one cluster contain value 1 rest will be zero.

   In general, for *n* entities

   $A_i \otimes B_i \otimes C_i = 1$ where $1 \leq i \leq n$

   Where$\otimes$ is exclusive or operator.
3. Each partition contains at least a single 1 and maximum *n*-2 1's.
4. Decimal value of second partition is always less than first partition.
5. Decimal value of third partition is always less than second partition.
6. Decimal value of first combination (tuple) of first cluster always starts with the value of power(2, *n*)-4
7. Decimal value of last combination (tuple) of third cluster always is power (2, *n*-2)-1.

4.2.4 Algorithm – 3-BitPartition

Algorithm 4.1 3-BitPartition

Input: Number of elements to be partition

Output: Bitwise generation of 3 partition

1. Read *n*
2. Initialize *i* = pow (2, *n*) -1
3. **for** *a* = pow (2, *n*) - 4 to 1 **do**

4. j = Complement(a) & i
5. if ($j \geq i$)
6. break;
7. **for** $b = j$ to 1 **do**
8. **if** (no overlapping of bits between (a & b) and union of (a & b) is not equal to i) **then**
9. c = (complement of (bitwise or of a & b)) bitwise & with i)
10. **end if**
11. **if** (value of third partition is less than 1st and 2nd partition) **then**
12. print(dectobin(a), a, dectobin(b), b, dectobin(c), c))
13. **end if**
14. **end for**
15. **end for**

Description of Algorithm:

Algorithm 4.1 generate 3 partitions in the bit format. 3-Partitions are generated in the 3n bits. Line no. 2 generates first partition (column) of partition of a particular tuple (Constraint no. 1). Line no. 4 checks complement of a is greater than a, if yes then it decrements the value of i. This step removes any repetition of tuples. Line no. 7 find out possible value of second partition (column). Line no. 8 checks whether there is any clash of bits in first partition and in second partition (check for Rule no. 2). Line no. 8 also checks third partition should contain at least a single 1. Line no. 9 generates the third partition. Line no. 11 checks value of third partition is less than first and second partition. Subroutine dectobin(int) translates decimal number into a binary string which represents a partition.

4.2.5 Characteristics and Validation of Algorithm

Proposed algorithm is a new approach for the generation and representation of 3-Partition. Bit approach is far better than other approach. Bit operations are used to generate partition and to check for the constraints. So, reduces time complexity. In this algorithm no duplicates tuples are generated. The beauty of this algorithm is that if the first partition (column) of any tuple is not according to constraints then it does not generate second partition (column) and similarly if the generated second partition clashes with any with the first cell then it does

not generate third partition (column) of the tuple. So, it reduces overall running time of the algorithm.

Algorithm is validated by the number of combinations generated are equal to the Stirling number of the second kind. This algorithm is very basic algorithm which can be applied in any of the application which requires 3-Partitioning. The most beautiful thing is that running time can be further reduced by applying Constraints. Constraints will be specific to application.

4.3. 3-Clustering Problem

3-Clustering is partitioning of O into 3 clusters C_1, C_2 and C_3 such that no cluster is empty, any pair of clusters has an empty intersection and the union of all clusters is equal to O. Minimum sum of Diameter for 3-Clustering is sum of diameter of 3 Cluster is minimum. Mathematically it can be defined as Minimum($d(C_1)+d(C_2)+d(C_3)$), where $d(C)$ is diameter of cluster C.

4.3.1 Methodology

In this method Clusters are represented in the form of bits. For the 3 clustering of n-entities 3i bits are required. Each of the n bits represents one cluster. When 3-Cluster is represented in bit form, a specific pattern is generated for 3-Partition for any value of n. In this pattern first n consecutive bits represent 1st Cluster (Partition), next n consecutive bits represent 2nd Cluster (Partition) and last n consecutive bits represent 3rd Cluster (Partition). If an entity O_i belongs to Cluster (Partition) C_i then in the i^{th} value of O_i is 1 otherwise it is zero. Let us consider a set of 5 entities $U = \{a, b, c, d, e\}$. Let us consider a Clustering Sample: $C_1 = \{a, b, c\}$, $C_2 = \{e\}$ and $C_3 = \{d\}$

Cluster 1					**Cluster 2**					**Cluster 3**				
a	*b*	*c*	*d*	*e*	*a*	*b*	*c*	*d*	*e*	*a*	*b*	*c*	*d*	*e*
1	1	1	0	0	0	0	0	0	1	0	0	0	1	0

Figure 4.3: Bit representation of 3-cluster

From the fig 4.3 bit representation of C_1, C_2 and C_3 is as

$C_1 = \{a, b, c\} = 11100$

$C_2 = \{e\} = 00001$

$C_3 = \{d\} = 00010$

4.3.2 Rule-Set for 3-BitClustering

1. Total number of 1's in 3*n* bits is always equal to *n* (so rest 2*n* bits are always zero).
2. Among the three clusters, on the same bit position exactly one cluster contain value 1 rest will be zero.

   In general, for *n* entities

   $A_i \otimes B_i \otimes C_i = 1$ where $1 \leq i \leq n$
3. Each cluster (partition) contains at least a single 1 and maximum n-2 1's.
4. Decimal value of second cluster (partition) is always less than first partition.
5. Decimal value of third cluster (partition) is always less than second partition.
6. Decimal value of first combination (tuple) of first cluster always starts with the value of power(2, *n*)-4
7. Decimal value of last combination (tuple) of third cluster always is power (2, *n*-2)-1.
8. The Minimum Sum of Diameter for 3-Clustering is always less than or equal to the third Maximum Dissimilarity value.

4.3.3 Algorithm

Algorithm 4.2 MinDiameter3-cluster

Input: A Matrix of *n* Elements

Output: Value of minimum sum of diameter for 3-clustering and entities for first and second maximum dissimilarity value

1. Find the Edge having maximum dissimilarity maxedge(*D*)
2. Let O_p and O_q be the entity having first maximum dissimlarity
3. Store(O_p ,O_q) Store entities in a array
4. Partition the elements in two cluster C_1 and C_2
5. Let $O_p \in C_1$ such that $|C_1| = n - 1$ and $O_q \in C_2$ such that $|C_2| = 1$

6. Calculate Minimum Sum of Diameter M_1 = MSD(C_1, C_2)
7. Let $O_q \in C_1$ such that $|C_1| = n - 1$ and $O_p \in C_2$ such that $|C_2| = 1$
8. Calculate Minimum Sum of Diameter M_2 = MSD(C_1, C_2)
9. M_3 = Min(M_1, M_2)
10. Find the Edge having second maximum dissimilarity maxedge(*D*)
11. Let O_l and O_t be the entity having second maximum dissimlarity
12. Store(O_l ,O_t) Store entities in a array
13. Partition the elements in three cluster C_1 , C_2 and C_3 such that
14. $|C_1| = n - 2, |C_2| = 1, |C_3| = 1$
15. If ($O_l \in C_1$ and $O_t \in C_3$) then $|C_1| = n - 2$ and $|C_3| = 1$
16. Calculate Minimum Sum of Diameter M_1 = MSD(C_1, C_2, C_3)
17. If ($O_t \in C_1$ and $O_l \in C_3$) then $|C_1| = n - 2$ and $|C_3| = 1$
18. Calculate Minimum Sum of Diameter M_2 = MSD(C_1, C_2, C_3)
19. M_4 = Min(M_1, M_2)
20. MSD = Min(M_3, M_4)
21. Store MSD in array
22. Return

Description of Algorithm:

Algorithm 4.2 find out first maximum dissimilarity and place them into different Clusters and finds a cluster that is having minimum value of sum of Diameter. Line no 1 to 3 returns pair of entities having first maximum dissimilarity and stores in an array. Return pair of entities can be put in two different ways in clusters. There are two possible combination (1) Entity $O_p \in C_1$ and $O_q \in C_2$ and (2) Entity $O_p \in C_2$ and $O_q \in C_1$. Line no. 4 to 9 compute cluster from the two-possibility having minimum sum of diameter. Line no. 10 to 12 return pair of entities having maximum dissimilarity and return pair of entities are stored in an array. Line no. 13 to 18 compute cluster for the two possibilities (for Second Entity Pair) for minimum sum of diameter. Line no 19 to 21 find out the minimum value of minimum sum of diameter and stores in the array.

Algorithm- 3Bit Clustering

Input: Number of entities to be clustered in three clusters (n) and a n * n Dissimilarity matrix (D)

Output: Minimum sum of diameter and corresponding three clustering pattern.

Variable: MCDBC(Minimum sum of diameter bit clustering) is an array that holds the vale return by algorithm BasicMinDia(). MCDBC holds five values, first values is minimum sum of diameter, next two values are entities of first maximum dissimilarity and last next two values are entities of second maximum dissimilarity

Algorithm 4.3 3-clustering for MSD

Input: A Matrix of *n* Elements

Output: Minimum sum of diameter and corresponding three clustering pattern

```
1.  int *M = BasicMinDia(D)
2.  Store the first maximum dissimlarity into reject list, RejectList(M[1], M [2])
3.  Store the second maximum dissimlarity into reject list, RejectList(M[3], M[4])
4.  Get the minimum values of 3-clusteringMSDC, MSD = M[0]
5.  for C1 = pow (2, n) – 4 to 1 do
6.      i = complement(C1)
7.      if (i > C1)
8.          break;
9.      end if
10.  if (FaultyVertex(C1)) then
11.   add FaultyVertex(C1) to RejectList()
12.  end if
13.    for C2 = i to 1 do
14.      if (FaultyVertex(C2)) then
15.          add FaultyVertex(C2) to RejectList()
16.      end if
17.      if ((C1 & C2 )  && (C1 || C2 )!= n-1 ) then
18.          C3 = (C1 || C2 )       // Generate 3rd Cluster
19.          if (value of C3 is less than C1 & C2) then
20.              D = Diameter(C1) + Diameter(C2) +Diameter(C3)
21.                if ( D <  MSD) then
22.                    MSD = D;
23.                     Cluster1 = C1;
24.                     Cluster2 = C2;
25.                     Cluster3 = C3;
26.                end if
27.          end if
28.      end if
```

29. **end for**
30. **end for**

Description of Algorithm:

Algorithm 4.3 Above 3BitCluster algorithm generates 3 Clusters in the bit format. 3- Clusters are generated in the 3n bits. Line no. 1 computes minimum value of minimum sum of diameter according to Algorithm 4.2. Line no. 2 and 3 add entities of first & second maximum dissimilarity value to the Reject List. Line no. 6 to 9 finds complement of C_1 and checks for the constraint no. 4. Line no. 10 to 11 check and add the faulty pair of entities to Reject Pair Entity List. Line no. 13 generate second cluster. Line no. 14 to 16 check and add the faulty pair of entities to Reject Pair Entity List. Line no 17 checks whether there is any clash of bits in first cell and in second cell. It also checks third cell should contain at least a single 1(Constraint no. 3). Line no. 17 and checks for the constraint no. 3, if satisfies then Line no. 18 Generate third cluster. Line no. 21-25 computes minimum sum of diameter and set minimum value.

4.3.4 Results

Proposed algorithm is a new approach for the generation and representation of 3-Cluster. Bit approach is far better than other approach. Numbers of tuples generated are verified and are equal to the Stirling number of the second kind, so even a single combination is not missed. From the Table 4.1 and Figure 4.4 it is very clear that Bit-clustering approaching is far better than traditional brute force approach. As the number of entities increases, proposed method gives better result than traditional approach. Its time complexity is less because of bitwise operation and if it does not generate 2^{nd} cluster if 1^{st} cluster is not valid and it does not generate 3^{rd} cluster if 2^{nd} is not valid. So, propose 3-clustering algorithm takes less time to compute minimum sum of diameter for 3-clustering.

4.4 Conclusion

Partitioning problems plays an essential role in theoretical computer science and in computational complexity theory. We applied the concept of bit representation to generate an algorithm for partitioning problems. The

proposed method uses bit operator to generate partition. In this chapter, we proposed a 3-Partitioning algorithm. The proposed partitioning algorithm doesn't generate duplicate tuples and generates only 3-partition tuples. Applying constraints specific to the application on this algorithm can further reduce the time complexity of the algorithm. In this chapter, an algorithm is proposed for the Minimum sum of diameter for 3-Clustering. In this method, bit operations are used to generate clusters and to check for the feasibility of constraints. In this algorithm, no duplicates tuples are generated. If the first cluster (column) of any tuple is not according to constraints, then it does not generate the second cluster (column). Similarly, if the bit pattern of generated second cluster clashes with the first cluster, the algorithm does not generate third partition (column) of the tuple. The proposed algorithm takes less time than the traditional 3-Clustering algorithm.

Table 4.1: Running time of BitClustering & Brute Force for 3-Clustering

No. of Entities	**Time taken by Bit Clustering (in second)**	**Time taken by Brute Force (in second)**
3	0.001	0.001
4	0.001	0.001
5	0.001	0.001
6	0.001	0.002
7	0.001	0.002
8	0.001	0.002
9	0.002	0.003
10	0.003	0.003
11	0.015	0.085
12	0.041	0.264
13	0.11	1.293
14	0.369	4.189

15	1.225	13.55
16	4.14	44
17	10.77	65
18	36	311
19	101	1012
20	610	3600

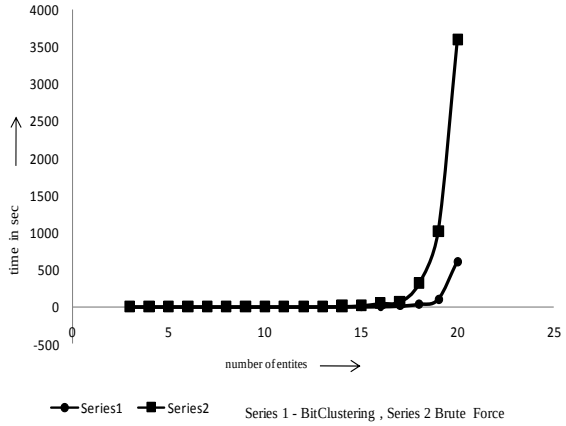

Figure 4.4: Comparison of Brute Force & 3-BitClustering approach

Chapter 5

Constraint Word Clustering Algorithm for Asymmetric Relationship

5.1 Introduction

Word Clustering is an important problem in web mining, natural language processing, automatic word classification, word sense, web analytics, computational linguists, and in parsing highly ambiguous syntactic structures [98, 99]. Word clustering is a technique for partitioning sets of words into subsets of similar words. Cluster of words can be identified on the basis of similarity between words or according to the affinities between words. A cluster comprises words that are sufficiently affine with each other. Words in the same cluster are highly affine and words in different cluster are less affine. Word clustering, is a useful approach for improving the performance of sentence retrieval, the more similar the words in each cluster, the better the performance of the retrieval system. Despite the usefulness of word clustering, accurately clustering the words remains a challenging task.

In the context of information retrieval, a new constraint word clustering is projected based on the paradigm of constraints for asymmetric relationship between words. Constraint word clustering approach is appropriate at discovering semantic relationship between words rather than discovering syntactic relationship between words. Affinity [99, 101] describes the quantitative relationship between words. An affinity describes a quantitative relationship between the two words and this in turn helps to identify the

clusters of words. A cluster comprises words that are sufficiently affine with each other. A first word is sufficiently affine with a second word if the affinity between the first word and second word satisfies one or more affinity criteria. Present research focuses on the clustering of words based on the finding of semantic relationship between words. Semantic relationships between words are modelled by identifying the constraints. Present research proposes a constraint clustering architecture and algorithm based on the different types of constraint associated between words.

5.2 Related Work

There have been a number of methods proposed in the literature that consider word clustering problem. Words with similar co-occurrence distributions is explored by Brown *et. al.* [102], it is based on class-based n-gram model in which words are clustered into word classes. Pereira *et al.* [103] present probabilistic membership of words and estimated a soft distributional clustering scheme for certain grammatical co-occurrences. In this strategy the conditional probability of a word is computed by taking advantage of observations of other words that act like this word in this context. A number of variant have been developed on this theme, using grammatical constraints such as part-of-speech, or morphological units such as lemma, or both [104]. Similarity based model are explored in [105-106] which avoids building clusters.

There are algorithms that automatically determine word classes without explicit syntactic or semantic knowledge. In [107] all words are gathered into a single class at the beginning of the procedure and are successively split to maximize the average mutual information of adjacent classes. In [108], a similar divisive clustering is proposed, based on binomial posteriori distributions on word co-occurrences. Text categorization can be achieved in various ways, in [109] Bag-of-Concepts is used to Improve the Performance of support vector machines. The impact of feature selection on document clustering is discussed in [110]. Hierarchical relationship and associative relationship, is a important in automatically building a thesauri or in finding associative relationship between words. Identification method [111] based on

co-occurrence analysis computer the hierarchical relationships between words. Our constraint word clustering method has an advantage over non-constraint clustering algorithm that it extracts background knowledge and guides the algorithm clustering and makes it more suitable for practical use.

5.3 Methodology

5.3.1 Affinity Computation and Modelling Based on Co-occurrence

Co-occurrence means coincidence or, frequent occurrence of two terms from a text corpus alongside each other in a certain order. Word co-occurrence in this linguistic sense can be interpreted as an indicator of semantic proximity. The global co-occurrence is an absolute or un-normalized metric. For the purpose of comparing term co-occurrences between different queries and sets of retrieved documents, co-occurrence is normalizing within a practical scale. So, co-occurrence values are normalized in the range of practical scale from 0 to 1.

Definition 9. The affinity [112] between any two words w_a & w_b is defined as the ratio of the number of co-occurrence that include both terms w_a and w_b over the maximum of either the number of co-occurrence contexts that include w_a or the number of co-occurrence contexts that include w_b . The Affinity is given by the following formula:

$$\text{Affinity}(w_a \cap w_b) = \frac{P(w_a \cap w_b)}{\max(P(w_a), P(w_b))} \tag{31}$$

Definition 10. The directional affinity [112] between word w_a & w_b is defined as the conditional probability of observing word w_b, given that word w_a was observed in a co-occurrence context. Directional affinity is used to describe the importance of word of word w_a with respect to word w_b. The directional Affinity (DAffinity) is given by the following formula:

$$\text{DAffinity}(w_a \cap w_b) = \frac{P(w_a \cap w_b)}{P(w_b)} \tag{32}$$

Definition 11. Average directional affinity [112] of a term w_a is the average of the directional affinity of a word with all other words in the co-occurrence contexts. The average directional Affinity (ADAffinity) is given by the following formula:

$$\text{ADAffinity}(w_a) = \frac{\sum_{j=1}^{N} P(w_a \cap w_j)}{N} \tag{33}$$

Definition 12. Differential directional affinity [112] of a term w_a is the difference of directional affinity w_a and average of the directional affinity of word w_a. Differential directional affinity (DiffDaff) is used to normalize the affinity of word with respect to other words.

$$\text{DiffDAff}(w_a) = \text{Affinity}(w_a \cap w_b) - \text{ADAffinity}(w_a) \tag{34}$$

5.3.2 Constraint Word Clustering

Wagstaff *et al.* [18] introduced constraints in the area of data mining research. Constraints provide guidance about the desired partition and make it possible for clustering algorithms to increase their performance. There are two types of constraints that were termed as must-link constraint and can-not link constraint. In must-link (ML) constraint two instances have to be in the same group, ML(*a*, *b*) symbolize instance *a* and *b* to have be in the same group. In cannot-link (CL) constraints two instances must not be placed in the same group, CL(*a*, *b*) symbolize instance *a* and *b* to have be in the different group. Let us consider words w_a and w_b, wcloud(w_a) and wcloud (w_b) are their respective word cloud.

Must Link Constraint: If wcloud(w_a) and wcloud(w_b) are similar then there exist a ML(w_a, w_b) constraint. It is represented in Boolean formulation as: $\text{ML}(w_a, w_b) \Rightarrow w_{ak} \wedge w_{bk} = 1$, where, w_{ak} means word w_a belong to k^{th} cluster.

Can-not Link Constraint : If wcloud(w_a) and wcloud (w_b) are not similar then there exist a CL(w_a, w_b) constraint. It can be represented as a Boolean formulation as: $\mathrm{CL}(w_a, w_b) \Rightarrow w_{ai} \wedge w_{bi} = 0$ where, w_{ai} means word w_a belong to i^{th} cluster and $i \neq j$.

5.4 Word Clustering Architecture

Constraint word clustering architecture contains two main components: knowledge matrix component and clustering component as shown in Figure 5.1. In this architecture, knowledge matrix component facilitates the building of affinity knowledge matrix based on the characteristics of source corpus. By varing the similarity criteria/measure different affinity knowledge matrix can be generated according to the need. Clustering component identify and generate the constraints and produces word cluster. Description of word clustering architecture is as follows:

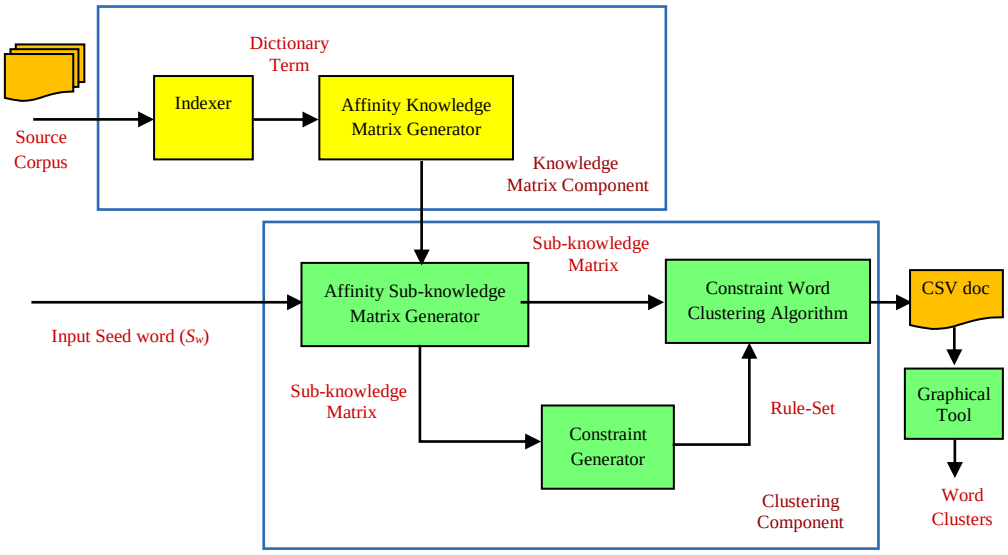

Figure 5.1 Architecture of constraint word clustering

Indexer: To search large amounts of text quickly it is required to convert the text into a suitable format that allows searching text rapidly. For this purpose, a suitable data structure inverted index table is used. Indexer is the component that builds the inverted index table from the source corpus. It eliminates the slow sequential scanning process of the text. Given a corpus $C =$

$\{D_1, D_2, \ldots, D_p\}$ containing p text documents, indexer takes this corpus as an input, identifies the dictionary terms, eliminates the stop words and builds the inverted index table *T*.

Affinity Knowledge Matrix Generator: Affinity knowledge matrix generator builds an affinity knowledge matrix using Algorithm 5.1. It is called knowledge matrix because it contains knowledge of whole corpora. It contains information that identifies how two words are closely associated. If inverted index table *T* contains information about *N* words, then corpus matrix generator builds a matrix of size *N* * *N*. It can be called as global knowledge affinity matrix corresponding to corpora *C*. It takes inverted index table *T* as the input; find out the affinity between each pair of every term of inverted index table. If size of inverted index table *T* is *N* terms then corpus matrix generator generates an *N* * *N* directional affinity knowledge matrix. Directional Affinity between each pair of term is calculated using (2). Affinity knowledge matrix represents the knowledge of corpora; it represents how the words are inter-related, degree of closeness between words.

Affinity Sub-knowledge Matrix Generator: This component takes input as a seed word and find out the cloud of words which are highly affine with seed word. Let us assume that size of word cloud is *n*. For each word of the cloud their respective word clouds are generated. From this word cloud sub-knowledge matrix is generated of the size *n* * *n*.

Constraint Generator: There are mainly two types of constraints Must Link (ML) constraint and Can Not (CL) constraint. If two objects are associated with ML constraint then they will belong to same group or class. On the other hand, if two objects are associated with CL constraint then they will belong to different group or class. Constraint generator performs analysis of sub-knowledge matrix and find out the word cloud which are exactly similar. If the directional average affinities of two clouds are equal then two clouds having same properties and they have same affinity behavior. If two word clouds are similar then there exist a must link constraint between them. Must link

constraint between any two words, enforce that the two words belong to the same cluster. Based on the analysis of knowledge matrix and word cloud following constraints are investigated.

Constraint Word Clustering Algorithm: Constraint word clustering algorithm find words which are highly affine with seed words and cluster them in such a manner that words in the same cluster are highly affine and words in the different cluster are less affine. In the proposed algorithm, existing *k*-means clustering algorithm is modified using concept of constraints. In the proposed constraints word clustering algorithm there are two major modifications, first modification is that words are assigned to the centroid (clusters) according to constraints and affinity values. Second modification is that centroid of cluster is one word, it is not like mean of the cluster like in *k*-means. In the word assignment step of constraint word clustering algorithm, if a word that belongs to RuleML and assigned to cluster C_i, then all others words of RuleML are also moved to cluster C_i. Similarly, if a word that belongs to RuleCL and assigned to cluster C_i, then all others words of RuleCL will not assigned to cluster C_i. Hence constraint clustering word algorithm gives good quality of clusters.

5.5 Investigation and Generation of Constraints and Rulesets

5.5.1 Investigation of Properties and Constraints Generation Based on Word Cloud

Following word cloud properties are investigated:

Property 1. Symmetric Property

if wcloud(w_i) = wcloud(w_j) then wcloud(w_j) = wcloud(w_i) then it gives constraint: ML(w_i,w_j)

Property 2. Transitive Property

if wcloud(w_i) = wcloud(w_j) and wcloud(w_j) = wcloud(w_k) then wcloud(w_i) = wcloud(w_k) then it gives constraint: ML(w_i,w_j, w_k)

Property 3. Implicative Property

if wcloud(w_i) = wcloud(w_j) and if wcloud(w_i) = wcloud(w_k) then wcloud(w_j) = wcloud(w_k) then it gives constraint: ML(w_i,w_j, w_k)

5.5.2 Constraint Generation Based on Association between Words

Two words are said to be associated if they are having some affinity value between them. Words are associated in either in one direction (forward or backword) or in both direction or not at all. Constratints are investigated on the basis of association between words as follows:

1. *Weak Association*(*One way association*): In weak association either the word w_a is associated with w_b or word w_b is associated with w_a. In weak association induces can not link constraint : CL(w_a, w_b)

2. *Strong Association*(*Two way association*): In strong association w_a is associated with w_b and word w_b is also associated with w_a. Mathematically, Aff(w_a, w_b) $\neq 0$ and Aff(w_a, w_b) $\neq 0$, it induces no constraint.

3. *Zero Association*(*no association*): In strong association w_i is not associated with w_j and word w_j is also not associated with w_i. Mathematically, Aff(w_a, w_b) = 0 and Aff(w_a, w_b) = 0. Zero Association induces can not link constraint in forward as well as in backward direction: CL(w_a, w_b) and CL(w_b, w_a)

5.5.3 Rule Set for Generation for ML and CL Constraints

In this Rule set are generated from the ML and CL constraint, rule set guide the constraint word clustering to obtain the desired partition.

Rule Set for Generation for ML Constraints

If w_a is the common word between any two ML constraints, then ML constraint can be merged to form a rule of must link constraint called as RuleML.

if ML_1= ML(w_a, w_b) and ML_2= ML(w_c, w_a) then Merge(ML_1, ML_2) => RuleML(w_a, w_b, w_c) = $w_{ai} \vee w_{bi} \vee w_{ci} = 1$

In general if a RuleML(w_1, w_2,…w_l) contains l words then it is given by the Boolean formula:

$w_{1k} \vee w_{2k} \vee ... \vee w_{lk} = 1$, where w_{lk} means word w_l belong to k^{th} cluster.

Rule Set for CL Constraints

If w_a is the common word between any two CL constraints, then CL constraint can be merged to form a rule of can-not link constraint called as RuleCL. if CL_1= CL(w_a, w_b) and CL_2= CL(w_c, w_a) then Merge(CL_1, CL_2) => RuleCL(w_a,w_b,w_c) = $w_{ai} \wedge w_{bj} \wedge w_{ck} = 1$, where $i \neq j \neq k$. In general if a RuleCL(w_1, w_2,…w_l) contains l words then it is given by the boolean formula: $w_{1k} \wedge w_{2k} \wedge ... \wedge w_{lk} = 0$, where w_{lk} means word w_l belong to k^{th} cluster and each word belongs to different cluster.

5.6 Word Clustering Algorithm

Algorithm 5.1 Affinity Knowledge Matrix

Input: A Corpus C consisting of documents such that $C = \{D_1, D_2,\ldots,D_p\}$, each document D is set of words.

Output: Affinity Knowledge Matrix (AKM) of size $N * N$, where N is the number of words and Inverted Index Table T.

1. Indexing of source corpus C to output inverted index table T, size of table T is N.
2. for i = 1 to N
3. for j = 1 to N
4. $$AKM[i][j] = DAff(w_i \cap w_j) = \frac{P(w_i \cap w_j)}{P(w_j)}$$

Description of Algorithm:

In step 1, source corpus C is indexed and inverted index table T is created. In step 2-4 affinity between each pair of word is calculated and a affinity knowledge matrix is generated.

Algorithm 5.2: Word Clustering

Input: Seed word – S_w, Threshold affinity value – δ, number of clusters – k, Affinity Knowledge Matrix- A*KM*, Inverted Index Table – *T*.

Output: *k* cluster of words

1. **if** $S_w \in T$ **then**
   C_w = WordCloud(AKM, S_w , δ)
2. **end if**
3. **for** i = 1 to n **do**
4. Wv_i = WordCloud(AKM, w_i)
5. **end for**
6. **for** i = 1 to n **do**
7. **for** j = 1 to sizeof(Wv_i) **do**
8. **if** $w_{ij} \notin C_w$ **then**
9. delete w_i from Wv_i
10. **end if**
11. **end for**
12. **end for**
13. ASKM = AffintySubKnowledgeMatrix(Wv_1, Wv_2, . , . , Wv_n)
14. Analyze ASKM, Let MLC = {ml_1, ml_2, ., ., ., ml_q} are q ML constraints are investigated
15. $\forall\, i, l, m,$ (where $1 \le i \le n, 1 \le l, m \le q$)
16. **if** $(w_i \in ml_l)$ and $(w_i \in ml_m)$ **then**
17. $rulelist = merge(ml_l, ml_m)$
18. **end if**
19. Assign k words as centroids c = {c_1, c_2, ., ., ., c_k} of k clusters C= {C_1, C_2, .., ., C_k}
20. **for** i = 1 to k **do**
21. $C_i = \{w_j : \max\left(\text{aff}\left((w_j, c_i)\right)\right) \forall\, j, 1 \le j \le n\}$
22. **if** $\left(w_j \in mlrulellist\right)$ **then**
23. $C_i = \{w_r : (w_r \in mlrulelist)\ \forall\, w_r\}$
24. **end if**
25. **end for**
26. **while** (old centroids and new centroids are same) **do**
27. **for** i = 1 to k do
28. η =sizeof(C_i)
29. **for** j = 1 to η **do**
30. $\hat{W}v_j$ = WordCloud(SKM, $\hat{w}_j$) where, ($\hat{w}_j \in C_i$)
31. **for** l = 1 to sizeof($\hat{W}v_j$) **do**
32. **if** $(w_{il} \notin C_w)$ **then**

33. delete w_i from $\hat{W}v_j$
34. **end if**
35. **end for**
36. **end for**
37. **end for**
38. CKM = ClusterSubKnowledgeMatrix($\hat{W}_{v1}$, $\hat{W}_{v2}$, . , . , $W_{v\eta}$)
39. **for** j = 1 to η **do**
40. $AA(w_j) = \sum_{m=1}^{\eta} CKM\,[j][m]/\eta$
41. $MeanCluster(C_i) = \forall w_j : (w_j \in C_i) \sum_{j=1}^{\eta} AA(w_j)/\eta$
42. $c_i = \{w_j : \min(|\text{MeanCluster}(C_i) - w_j|)\, \forall\, j, 1 \leq j \leq \eta\}$
43. **end for**
44. **end while**

Explanation of Algorithm:

In step 1 it is checked whether the input seed word belong to corpus or not. In step 2, word cloud is generated corresponding to seed word S_w. C_w contains all words which are affine with S_w and whose affinity value is less than δ. In step 3-5, word clouds are generated for all *n* belonging to C_w. In step 6-12, spurious words are deleted from each word cloud, spurious words are that word which does not belong to C_w or which are not affined with S_w. This step normalizes the size of each word cloud to *n*. In step 13, Sub knowledge matrix is generated from *n* word cloud of size *n* * *n*. In step 14, Sub knowledge matrix is analyzed and ML constraints are generated. In step 14-18, if a word belongs to more than one ML constraints then all ML constraints are merged to form a rulelist. If ML constraints are not mutually related then, they ML constraints will belong to different rulelists. In step 19, randomly *k* words are assigned as centroids of *k*-clusters. In step 21 words are assigned to their respective clusters based on their maximum affinity with the centroids. In step 22-23, if a word w_r is an element of to a *mlrulelist* and belongs to a cluster C_i then all other words of same rule list are assigned to cluster C_i and their status is updated to assigned so assigned words are not checked for comparison and assignment. Step 26-42 is executed until there is no change in the centroids of two consecutive iterations. In step 26-38, for each word of each cluster, word cloud is generated and after removal of spurious words Cluster-knowledge matrix is generated. In step 39-40 average affinity of each word in cluster is

calculated. In step 41, new mean of each cluster is calculated. In step 42, word w_j which is closest to the mean of the cluster C_j is assigned as the new centroid cluster C_j.

5.7 Conclusion

We proposed a method to identify and generate constraints between words that identify semantic similarity measure between words and word clouds. We investigated different types of association between words and identified constraints based on the investigated association between words. Moreover, a constraint-based word clustering algorithm is proposed. In the proposed approach, words clouds are compared rather than words, which extract semantic meaning of words in the respective group of words or in the respective affinity sub-knowledge matrix for the generation of constraints. Constraints provide guidance about the desired partition and make it possible for clustering algorithms to increase the accuracy of clusters generated. Proposed approach is applicable for symmetric as well as asymmetric relationship between words. Thus, constraint word clustering algorithm is useful extension to conventional word clustering algorithm.

Chapter 6

Approximation Constraint Clustering Techniques

6.1 Introduction

An approximate algorithm is a way of dealing with NP-completeness for the optimization problem. This technique does not guarantee the best solution. The goal of an approximation algorithm is to come as close as possible to the optimum value in a reasonable amount of time which is at the most polynomial time. Such algorithms are called approximation algorithm. In this chapter, concept of constraints is being introduced in min-cost *k*-cover problem to present a new constraint-based min-cost *k*-cover algorithm. We modify the $O(n^{881}.T(n))$ polynomial time exact algorithm of Gibson *et al.* [90] to acquire another $O(n^{\lambda}.T(n))$ polynomial time constraint-based algorithm, where $\lambda < 881$ and λ relies on the number of can-not link constraint.

The proposed technique for the problem of min-cost *k*-cover is achieves an optimal result based on the constraints. The present research identifies the can-not link constraints and apply these derived constraints in the algorithm for the reduction of distinct maximal disks and reduction of all enumerated subsets of the distinct maximal disks in minimum sum of radii problem. This in turn reduces the number of list entries of exact algorithm. The basis for the constraint technique is motivated by an observation that in any instance *I* of *k*-cover, the optimal outcome is at most the maximum radius r of ball $B(v, r)$ positioned at $v \in V$ in *I*. It infers that, in any instance there always exists a constraint that separates the optimal solution.

6.2 Euclidean min-cost *k*-cover problem

The Euclidean min-cost *k*-cover problem defined as follows. Given a metric d defined on a set V of points, we define the ball $B(v,r)$ centered at $v \in V$ and having radius $r \geq 0$ to be the set $\{ q \in V \mid d(v,q) \leq r\}$. In the minimum cost *k*-cover problem, we are given a set *P* of *n* points and integer *k* (*k* > 0). For $\kappa > 0$, computing a κ-cover for subset $Q \subseteq P$ is a set of at most κ balls covering all point of set *Q* and each ball centered at a point in *P*. The sum of the radii of balls is the cost of a set *D* of ball denoted by cost(*D*). In the Euclidean version, *P* is given as a set of points in some fixed dimensional Euclidean space R^l, and *d* is the standard Euclidean distance. In the metric version of the min-cost *k*-cover problem, we have *P* and *k* and the distance *d* between every pair of points in *P*.

6.3 Preliminaries

Research commence with the observation that for any instance *I* of *k*-cover, the solution value is at most the maximum radius *r* of ball $B(v,r)$ and optimal solution is decidedly separable. Thus, it permits to compute an optimal *k*-cover efficiently using constraints. We call a ball of zero radius as singleton ball. Similarly, we call a disc of zero radius as singleton disc.

Definition 13. A distinct maximal disc (DMD) is a disc if one cannot add any point to it without increasing its radius. Any solution can be reduced into one having only distinct maximal disc without increasing the cost. Thus, non-distinct maximal discs can be ignored to obtain optimal solution.

Lemma 3. *For any instance of I of k-cover, the optimal solution value is at most the maximum radius r of ball B(v, r) in I.*

Proof. Given any instance of *k*-cover, a solution $B = \{B_1, B_2, \dots, B_k\}$ consisting of *k* balls covering all instance of *I*. The cost of *B* is given by the following formula $\text{cost}(B) = \sum_{i=1}^{k} radius(B_i)$. Assume balls B_2, B_3,...,B_k as singleton balls. Assign randomly chosen $n-k+1$ points to ball B_1 and assign remaining $k-1$ points to $k-1$ balls such that each ball contains single point. Hence, $\text{cost}(B) = \text{cost}(B_1) = radius(B_1)$. Therefore the the optimal solution

value is at most the maximum radius *r* of ball *B*(*v*, *r*) in *I*.

Lemma 4. In the min-cost *k*-cover problem, the number of distinct maximal discs is at most n^2

Proof. Let us consider a set of points $P = \{v_0, v_1, \ldots v_l\}$ and $r_0, r_1, \ldots r_n$ are the sorted distance from the point v_0 in the ascending order. For any value of $1 < i \leq l$, consider a ball $B(v_o, r)$ of radius *r*, from the Figure 6.1 it is understood that $r_i \leq r < r_{i+1}$. It means that for $1 < i \leq l$, if $r_i \leq r < r_{i+1}$ then $B(v_0, r_i) = B(v_0, r)$. So, the only distinct maximal disc centered at v are $B(v, r_i)$ for $1 \leq i \leq l$ and v is center of $l \leq n$. Each point can be the center of at most *n* distinct maximal discs, distinct maximal discs also include disk of radius zero or singleton disk, and therefore there are at most n^2 distinct maximal discs. The figure of separated maximal discs for the problem of min-cost *k*-cover is at most n^2 (Lemma 3). This gives critical benefit in the stated problem. Entire subclasses of the unrelated disc with size at most *l* can be enumerated in time $O(n^{2l})$. For fixed value of *l*, enumeration of subclasses is calculated in polynomial run.

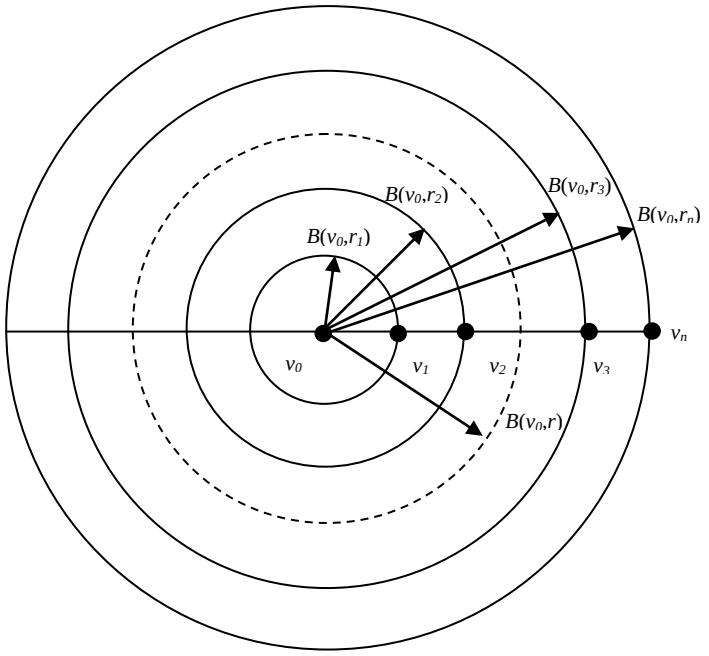

Figure 6.1: Distinct maximal discs

6.4 Constraints based min-cost *k*-cover Approach

Wagstaff *et. al.* [18] presented imperatives in the zone of information mining research. There are two sorts of limitations viz. "must-link" constraint and "can-not link" constraint. Two instances must be at similar gathering in must-link (ML) constraint. ML(*a*, *b*) symbolize occurrence *a* and *b* to should be in the similar group. Two instances must be in the different cluster in cannot-link (CL) constraint. CL(*a*, *b*) symbolize appearance of *a* and *b* should be in the distinctive group. Algorithm 1 finds can-not link constraints in any instance *I* of *k*-cover problem.

6.5 Constraint Algorithm

Algorithm 6.1 takes input as any instance of *I* of min-cost *k*-cover problem. It computes an initial feasible solution in accordance to Lemma 1. Can-not link constraints are investigated and generated for individually ball $B(o,r)$, where $o \in O$ and *r* is the radii of the ball. It returns a set of can-not link constraints.

Algorithm 6.1 Constraintcover(*I*)

Input: Instance of min-cost *k*-cover problem

Output: Set of can-not link constraints

1. Find an initial feasible solution and compute the cost of initial feasible solution(cost(*D*))
2. For $\forall\ v$ such that $v \in V$
3. For each ball *B*(*v*, *r*) and for $\forall\ w$ such that $w \in V$, such that r = |*vw*| ($v \neq w$)
4. If *r* > cost(*D*) then associate can-not link constraint between *v* and *w:* CL(*v*, *w*)
5. *S* = *S* U CL(*v*, *w*)
6. return *S*

Theorem 8. Maximum number of can-not link constraints generated in any instance *I* of *k*-cover problem is order of $O(n^2)$.

Proof. We have instance *I* of *k*-cover problem consisting of *n* points. For any two points *p*, *q* can-not link constraint exist in $B(v,r)\ if\ r > |pq| >$ Icost(D). Formally it can be stated as $\exists\ \mathrm{CL} - \mathrm{constraint},$ If ($|pq| >$

Icost$(D))$, $\forall\, p, q: p, q \in V$. Point p and q can be any point from the *V*, so maximum n^2 combination can be possible. This is similar to the finding the distinct maximal discs. The number of distinct maximal discs is at most n^2 (by Lemma 4). Therefore, Maximum number of can-not link constraints is order of $O(n^2)$.

Theorem 9. Minimum number of can-not link constraints generated in any instance *I* of *k*-cover problem is order of $O(k)$.

Proof. For k =1, all points of *I* are covered by a single ball $B(v, r)$. Assume that a can-not link constraint CL(p, q) exist between any two points p, q of ball $B(v, r)$, then p, q together $\exists\, CL(p, q) |\ p, q \in B(v,\ r)$then ($p$ and $(n-2)$points) $\in B_1(v,\ r_1)$ and point $q \in B_2$ can-not Then, $n-1$ points belong to $B_1(v, r_1)$ and point q belong to singleton disk B_2. In this manner if there are $k-1$ can-not link constraints then *k* balls can cover *n* points, $n-k+1$ points are covered by a ball $B_i\,(v, r_i)$ and rest of the *k*-1 points are covered by *k* singleton disks. Solution obtained in this way is a feasible and optimal solution (by Lemma 1).

Theorem 10. Number of discs in reduced distinct maximal discs is always less than distinct maximal discs, $|D_R| < |D|$.

Proof. Count of different maximal discs is greatest of n^2, and the optimal solution contains discs that belong to the group of several maximal discs. Let D_r denotes the compact maximal distinct discs. There are always exist at least a can-not link constraint in any instance of min-cost *k*-cover problem (Theorem 9). If a can-not link constraint belongs to distinct maximal disc then that distinct maximal disc will not be part of optimal solution. So, with the help of can-not link constraints, it can be verified whether a distinct maximal disc will be a part of the canonical optimal solution or not. Hence, can-not link constraints reduce the number of distinct maximal cluster. Let *S* denote set of can-not link constraints and α denoted the cardinality of set *S*, then the maximum value of α is $O(n^2)$ (by Theorem 8). Applying α constraints ($O(n^2)$) on the distinct maximal discs $D(O(n^2))$ reduces *D* by the significant factor, hence $|D_R| < |D|$.

Corollary 1: *A can-not link constraint clearly separate the optimal solution.*

6.6 Constraint based min-cost *k*-cover algorithm

Assuming a sample of the Euclidean min-cost *k*-cover problem which comprises of an arrangement of points *O* on the plane alongside a whole number *k*. Euclidean separation is the distance between any combination of points on the plane. Considering the set of distinct maximal discs *D* with radius |*pq*| and center $p \in O$ for some $q \in$ U. *D* incorporated the disk of radius zero, therefore, $|D| = n^2$.

Gibson et al. approach [90]: In a balanced rectangle, the proportion of width to length is no less than 1⁄3. This method utilizes "balanced rectangle" to define the sub-task. For a rectangle *R*, a separator is a line opposite to the lengthier side, and it intersects in the middle third of its more extended edge of the R. The computation begins with a rectangle R_0 containing every one of the points and divides into two littler rectangles by picking a filter line and resolves the sub-issues recursively. The "vertical line or horizontal line" is termed critical in the event that it either experiences an entity $p \in P$ or if it is tangent to any disk in *D*. Every single vertical line between two back to back basic vertical lines converge a similar arrangement of discs. Therefore, there are just $\Theta(n^2)$ vertical or horizontal lines as separators. To get an optimal solution, it is required to consider just $|T| \leq \beta = 424$. It signifies that span of the dynamic programming list is $O(n^{2\beta+5})$, which is polynomial limited.

Our Constraint algorithm: A set *S* of can-not link constraints are generated using Algorithm 6.1. The constraint min-cost *k*-cover algorithm takes rectangle R, a whole number $\kappa \geq 0$, a subset $T \subseteq D$, a set of constraint *S* and has a iterative algorithm $\mathrm{DC}(R, \kappa, T, S)$. It computes a most favourable solution utilizing highest κ discs from the arrangement of arguments in $Q = \{q: q \in (P \cap R)\}$. A can-not link clearly separate the optimal solution (by Corollary 1). This implies that a separator is simulating like a can-not link constraint and diving the problem into sub-problem and tackling it recursively. The procedure

invokes DC$(R_0, k, \emptyset, S)$ to locate finest cover for P by applying α constraints. The estimation of the sub-task for a iterative call is put away in a runtime programed list List$[P \cap R, \kappa, T]$. In our calculation beginning strides are essential initialization step and it stays same as our algorithm takes after the means as of algorithm [13]. Numbers of separators are directly proportional to the number of distinct maximal disk. Our based constraint approach uses the reduced distinct maximal disk instead of the distinct maximal disk so that some separators will be decreased. In subsequent steps, can-not link constraints are applied on D to obtain reduced distinct maximal discs (D_r), In further steps D_R is used for the computation of entries in the list.

Algorithm 6.2 Constraint Clustering DC$(\boldsymbol{R_0}, \boldsymbol{k}, \emptyset, \boldsymbol{S})$

Input: Rectangle R, whole number $\kappa \geq 0$, subset $T \subseteq D$, set of constraint S

Output: minimum cost k-cover of n points

1. if event List$[P \cap R, \kappa, T]$ already exist return, else, generate it.
2. Consider $Q = \{q: q \in (P \cap R)\}$ (Assuming T doesn't contain the q)
3. if $Q = \emptyset$ then List$[P \cap R, \kappa, T] \leftarrow \emptyset$ else leave from the present point
4. On the chance that $\kappa = 0$, assign List$[P \cap R, \kappa, T] \leftarrow \{I\}$ Echo infeasible cover and return from the present step
5. In the situation $|Q| = 1$, let List$[P \cap R, \kappa, T]$ is arrangement of singleton disk and return
6. Invoke procedure compress(R) (to acquire a "balanced rectangle" R' containing ($P \cap R$)). $|L(R')|$ is likewise lessened in light of the fact number of separators are directly proportional to the cardinality of distinct maximal disc. Constraint lessens the cardinality of distinct maximal disc along these lines $|L(R')|$ likewise diminishes.
7. Initialize a cover $D' \leftarrow \{I\}$. (I is a dummy disc of ∞)
8. Considering all ranges of partitioning lines $l \in L(R')$ do
9. Assign $D_R = D$
10. $\forall$ disk d such that $d \in D$ (for every distinct maximal disk that belongs to set D)
11. Apply can-not link constraints on distinct maximal discs to obtain reduced distinct maximal discs (D_r), $|D_R| < |D|$ (by Theorem 3)
12. if there exists a CL constraint among a, b such that points $a, b \in d$ then $D_R = D_R - d$
13. if $((D_0 \subseteq D_r)$and $|D_0| \leq 12))$, for all values of D_0 intersected by l do
14. for all values of κ_1, $\kappa_2 \geq 0$ with the goal that $(\kappa_1 + \kappa_2 + |D_0|) \leq \kappa$ do

15. Assume ($R_1$and R_2) are two rectangular partitions gained by dividing R'through l consider $T_1 = \{D_r \in (T \cup D_0) | D_r$ crosses $R_1\}$ and $T_2 = \{D_r \in (T \cup D_0) | D_r$ crosses $R_2\}$
16. if $(|T_1| \leq \lambda < \beta)$ and $(|T_2| \leq \lambda)$ at that point
17. Invoke DC(R_1, κ_1, T_1, S) and DC(R_2, κ_2, T_2, S) in an iterative manner
18. If cost$(D_0 \cup$ List$[P \cap R_1, \kappa_1, T_1]) \cup$ List$[P \cap R_2, \kappa_2, T_2]) < cost(D')$ then at that point change $D' \leftarrow D_0 \cup$ List$[P \cap R_1, \kappa_1, T_1] \cup$ List$[P \cap R_2, \kappa_2, T_2]$
19. Allocate List$[P \cap R, \kappa, T] \leftarrow D'$
20. Return

Running Time: The execution duration of a procedure DC$(R_0, k, \emptyset, S)$ remains limited by number of entries in the list. For a balanced rectangle *R*, every entry in the list is recorded by a collection of points $(P \cap R)$ for some *R*, a $(\kappa \leq k)$ and a set $T \subseteq D$ in such a way that $|T| \leq \beta = 424$. Proposed constraint based approach reduces the β hence, running time also reduces. Number of disk inside *R*, crossed by a separator is maximum 12 (Lemma 2.1 in [90]). Number of disk intersecting from outside of is 40(Lemma 2.2 in [90]). Assuming total number of separator (horizontal and vertical) is at most 32. In this manner, there are at most 32*12 = 384 discs approaching from the separators. Total number of disc intersecting from outside and inside the rectangle are β =384+40=424. Enumerating all possible set of β takes $n^{2\beta}$ = n^{2*424} =n^{848}. Some other steps are taking a time n^5, giving total computation n^{853}. Direct recursion takes n^{28} time and n^{24} time for D_0, so total running time of algorithm is $n^{853+28} = n^{28} * n^{2*(32*12+40)+5}$. So, the value of β is = 28 +2*(32*12+40)+5 = 881. In other words, β *is* proportional to (28 * 2*(number of separator * number of internal disk intersecting *R* + number of external disk intersecting *R*)+ 5). Number of separator, the number of internal disk intersecting *R* and number of external disk intersecting *R* are dependent on number of distinct maximal disks. Our constraint based algorithm uses the reduced distinct maximal disks (D_r) instead of distinct maximal disks (*D*) so β will be reduced to λ. It is not hard to see that applying n^2 constraints reduces number of distinct maximal discs, which thus enhances the performance of the algorithm.

6.7 Conclusion

Constraint based algorithm has attracted researchers due to its characteristic, of strengthening the performance of algorithms by adding constraints as a parameter. In this context, usage of a constraint based approach to min-cost *k*-cover problem improves the bound of the algorithm. The research portrays how constraint based algorithm is convenient and yields better empirical results compared to non-constraint algorithms for the min-cost *k*-cover problem. In the min-cost k-cover problem, the number of entries stored in a list is bounded by $O(n^{881})$. In this research, we identify can-not link constraints and proposed an algorithm that decreased the number of the records and the number of distinct maximal discs. The reduction of distinct maximal disks number enhances the performance of the algorithm by a noteworthy factor.

Chapter 7

Conclusion

7.1 Discussion

Minimum sum of diameter clustering and minimum sum of radii clustering has attracted researchers due to its special characteristic, of avoiding dissection effect. Therefore, an extensive number of applications are adapting MSDC and MSRC technique. This appears to be a quite novel approach in the area of data mining to find minimum sum of diameter and radii clustering via constraints. This thesis is set out to apply constraint technique for minimum sum of diameter and minimum sum of radii clustering. Our main focus has been investigating different constraint based methods in order to enhance the clustering algorithm results. Motivation has been on the one hand to reduce dimensionality in order to keep running times low and on the other to enhance clustering results.

In this thesis, the results of the study are represented as a comparative analysis of clustering algorithms in tabular form. Two tables are produced as an outcome of the literature review, presented tables provide comprehensive evaluation, and drew a comparative analysis of algorithmic on the following factors: clustering criteria, approach, constraints, assumptions, issues, and time complexity. Various exact and approximation clustering algorithms for Euclidean, metric, and geometric versions of the MSDC problem and MSRC problem are investigated. The presented comprehensive review helps researchers to identify the research gaps to develop more optimized algorithms.

The reduction of the clustering problem into SAT formulation plays an essential role in analyzing social networking problems. We presented a formulation for the reduction of 3-clustering to 3-SAT and *k*-clustering to *k*-SAT. The research portrays how constraint clustering for the minimum sum of diameter and radii is convenient and yields better empirical results compared to non-constraint clustering algorithms for the minimum sum of diameter and radii. Moreover, the research study is quite successful in presenting a new bit wise approach for the 3-clustering minimum sum of diameter algorithm. The research also explores syntactic relationship between words and finds significant increase in the accuracy of word clustering algorithm. Last but not the least, the present research reveals that constraint based min cost *k*-cover improves the bound min cost *k*-cover algorithm. Last but not least, the present research reveals that constraint based technique can be applied to approximation algorithms. The proposed constraint based method decreased the number of the distinct minimal disc, and the proposed algorithm improves the bound min cost k-cover algorithm.

7.2 Future Scope of the Work

This section briefly describes the areas that can fall in thesis scope that are quite possibly an extension for future scope. The research suggests that in order to obtain polynomial time algorithm for 3-clustering, Boolean approach can be extended by exploration of properties of tripartite graph. In the present research, can-not link constraints reduced the size of the table entries and reduced the time complexity of algorithm by a significant factor. It could be more interesting to find out more can-not link constraints and must-link constraints to reduce the number of call stored in table for min-cost *k*-cover problem. Present model of min-cost *k*-cover problem is based on minimum sum of radii criterion. The way the model is constructed could be also changed by using minimum sum of diameter criterion. If the number of distinct maximal cluster is bounded by polynomial time then polynomial time approximation scheme for MSDC can be modeled.

Bibliography

[1] Cormack R. M. (1971), A review of classification, Journal of the Royal Statistical Society, Series A, 134, 321–367.

[2] Hartigan J. A. (1975), Clustering algorithms, John Wiley & Sons, New York.

[3] Anderberg M. R. (1973), Cluster analysis for applications, academic, New York.

[4] Hansen P., Jaumard B., (1987), Minimum sum of diameters clustering, Journal of Classification, 4, 215–226.

[5] Gordon A. D. (1981), Classification: methods for the exploratory analysis of multivariate Data, Chapman and Hall, New York.

[6] Jain A., Dubes R. (1988), Algorithms for clustering data. Prentice-Hall, Inc., Englewood Cliffs, NJ.

[7] Raghavan P. (1997), Information retrieval algorithms: a survey. In: Saks M E ed. Proceedings of 8th ACM-SIAM Symposium on Discrete Algorithms (SODA'97), New Orleans, Louisiana, pp. 11–18.

[8] Duds R., Hart P. (1973), Pattern classification and scene analysis, Wiley-Interscience, New York, NY.

[9] Wang H., Pei J. (2008), Clustering by pattern similarity, Journal of Computer Science and Technology, 23(4), 481–496.

[10] Hansen P, Jaumard B. (1997), Cluster analysis and mathematical programming, Mathematical Programming, 79, 191–215.

[11] Xu R., Wunsch D. (2005), Survey of clustering algorithms, IEEE Transaction Neural Networks, 16, 645–678.

[12] Duran B. S., Odell P. L. (1974), Cluster Analysis: A Survey, Heidelberg: Springer-Verlag.

[13] Delattre M., Hansen P. (1980), Bicriterion cluster analysis, IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI, 2(4), 277–291.

[14] Xiong H. (1996), Some undecidable problems on approximation of NP

optimization Problems, Journal of Computer Science and Technology, 11(2), 126–132.

[15] Brucker P. (1978), On the complexity of clustering problems, In: Beckmann M, Kunzi H P eds. Optimization and Operations Research, Lecture Notes in Economics and Mathematical Systems, 157th edition, Heidelberg: Springer-Verlag, 45–54.

[16] Sahni S, Gonzalez T F. (1976), P-complete approximation problems, Journal of ACM, 23, 555-565.

[17] Karp R. M. (1972), Reducibility among combinatorial problems, Complexity of Computer Computations, Miller R E, Thatcher J W, New York: Plenum, 85–103.

[18] Wagsta K., Cardie C. (2000), Clustering with instance-level constraints, In: Proceedings of the Seventeenth International Conference on Machine Learning, pp. 1103-1110.

[19] Rao M. R. (1971), Cluster analysis and mathematical programming, Journal of the American Statistical Association, 66(335), 622–626.

[20] Hansen P., Delattre M. (1978), Complete-link cluster analysis by graph coloring, Journal of the American Statistical Association, 73(362), 397–403.

[21] Guénoche A., Hansen P., Jaumard B. (1991), Efficient algorithms for divisive hierarchical clustering with the diameter criterion, Journal of Classification, 8, 5–30.

[22] Charikar M., Panigraphy R. (2004), Clustering to minimize the sum of cluster diameters, Journal of Computer and Systems Sciences, 68(2), 417-441.

[23] Mishra N., Schreiber R., Stanton I., Tarjan R. E. (2007), Clustering Social Networks, Springer, Lecture Notes in Computer Science, 2007, 4863, 56-67.

[24] Cook S A. (1971), The complexity of theorem proving procedures. In: Proceeding of the 3rd Annual ACM Symposium Theory Computing, pp. 151–158.

[25] Garey M. R., Johnson D. S. (1979), Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman, San Francisco.

[26] Welch J. W. (1983), Algorithmic complexity: three NP-hard problems

in computational statistics, Journal of Statistical Computation and Simulation, 15, 17–25.

[27] Zuckerman D. (1993), NP-complete problems have a version that's hard to approximate, Structure in Complexity Theory Conference, In: Proceedings of the Eighth Annual, IEEE, pp. 305–312.

[28] H. Li, N. Abe (2002), Word Clustering and Disambiguation Based on Co-occurrence Data, Journal of Natural Language Engineering, 8(1), 25 – 42.

[29] Doddi S. R., Marathe M. V., Ravi S. S. *et al*. (2000), Approximation algorithms for clustering to minimize the sum of diameters, Nordic Journal of Computing, 7 (3), 185–203.

[30] Lev-Tov N., Peleg D. (2005), Polynomial time approximation schemes for base station coverage with minimum total radii, Computer Networks, 47, 489–501.

[31] Hopcroft J. E., Ullman J. D. (1979), Introduction to automata theory, languages and computation, Addison-Wesley, pp. 324- 325

[32] Kolen J. F. (2002), An on-line Satisfiability for conjunctive normal form expressions with two literals, Flairs -02, Proceedings, pp.187 – 191.

[33] Aspvall B., Plass M. F., Tarjan R. E. (1979), A Linear-time algorithm for testing the truth of certain quantified Boolean formulas, Information Processing Letters, 8, 121–123.

[34] Hopcroft J. E, Ullman J. D. (1969), Formal Language and their Relation to Automata, Addison-Wesley, Reading, Mass.

[35] Aho A. V., Hopcroft I. E., UIlman J. D. (1974), The Design and Analysis of Computer Algorithms Addison-Wesley, Reading, MA.

[36] Sahni S. (1974), Computationally related problems, SIAM Journal of Computing, 3(4), 262–279.

[37] Dasgupta S., Papadimitriou C. H., Vazirani U. V. (2006), Algorithms, McGraw-Hill Higher Education, pp. 244–245.

[38] Hammer P. L., Rudeanu S. (1968), Boolean Methods in Operations Research and Related Areas, New York: Spriniger-Verlag.

[39] Johnson S. C. (1967), Hierarchical clustering schemes, Psychometrika, 32, 241–254.

[40] Even S., Itai S., Shamir A. (1976), On the complexity of time table and multi-commodity flow problems, SIAM Journal of Computing, 5(4), 691–703.

[41] Schaefer T. J. (1978), The complexity of satisfiability problems. In: Lipton R J, Burkhard, Walter W A, Savitch J et al. eds. Proceeding of 10th Annual ACM Symposium Theory Computing, San Diego, California, USA, pp. 216–226.

[42] Tarjan R. E. (1992), Depth fist search and linear graph algorithms, SIAM Journal of Computing, 1(2), 146–160.

[43] Asano T., Bhattacharya B., Keil M. et al. (1988), Clustering algorithms based on minimum and maximum spanning trees. In: Proceeding of the 4th Annual symposium on Computational geometry, Urbana-Champaign, IL. ACM Press, pp. 252–257.

[44] Monma C., Suri S. (1989), Partitioning points and graphs to minimize the maximum or the sum of diameters. In: Proceedings of the sixth International conference on the Theory and applications of graphs, Wiley, New York.

[45] Hubert L. (1973), Monotone invariant clustering procedures. Psychometrika, 38(1), 47–62.

[46] Hershberger J. (1992), Minimizing the sum of diameters efficiently, Computational Geometry: Theory and Applications, Elsevier, 2, 111–118.

[47] Gabow H. N., Tarjan R. E. (1985), A linear-time algorithm for a special case of disjoint set union, Journal of Computing System Science, 30, 209-221.

[48] Zuckerman D. (1996), On unapproximable versions of NP-complete problems, SIAM Journal of Computing, 25, pp. 1293–1304.

[49] Hagauera J., Rote G. (1997), Three-clustering of points in the plane, Computational Geometry, 8(2), 87–95.

[50] Capoyleas V., Rote G., Woeginger G. (1991), Geometric clusterings, Journal of Algorithms, 12, 341–356.

[51] Ramnath S. (2002), Forewarned is fore-armed: dynamic digraph connectivity with lookahead speeds up a static clustering algorithm. Algorithm Theory-SWAT 2002, Lecture Notes in Computer Science,

2368, 220–229.

[52] Khanna S., Motwani R., Wilson R. H. (1996), On certificates and lookahead on dynamic graph problems. In: Clarkson K L ed. Proceeding of 7th ACM-SIAM Symposium, Discrete Algorithms, San Francisco, California, 222–231.

[53] Noman Z., Khan M. (2002), Performance of two algorithms in minimum sum of diameters clustering. In: 35th Annual Midwest Instruction and Computing Symposium, Iowa.

[54] Jain R., Chaudhari N. S. (2011), Formulation of 3-clustering as a 3-SAT problem, In: Proceeding of Fifth Indian International Conference on Artificial Intelligence (IICAI), India, 2011, pp. 465-472.

[55] Gonzalez T. F. (2007), Handbook of Approximation Algorithms and Metaheuristics (Chapman & Hall/Crc Computer & Information Science Series), Chapman & Hall/CRC, New York.

[56] Fowler R. J., Paterson M. S., Tanimoto S. L. (1981), Optimal packing and covering problem are NP complete, Information Processing Letters, 12(3), 133–137.

[57] Megiddo N., Supowit K. J. (1984), On the complexity of some common geometric location problems, SIAM Journal of Computing, 13(1), 182–196.

[58] Drezner Z. (1984), The *p*-centre problem-heuristic and optimal algorithms, The Journal of the Operational Research Society, 35(8), 741–748.

[59] Dyer M., Frieze A. M. (1985), A simple heuristic for the p-center problem. Operation Research Letter, 3, 285–288.

[60] Hochbaum D. S., Pathria A. (1997), Generalized p-center problems: complexity results and approximation algorithms, European Journal of Operational Research, 100(3), 594–607.

[61] Gonzalez T. F. (1985), Clustering to minimize the maximum intercluster distance, Theoretical Computer Science, 38, 293–306.

[62] Hochbaum D. S., Shmoys D. B. (1984), Powers of graphs: a powerful approximation technique for bottleneck problems. In: DeMillo R A ed. Proceeding of Symposium on the Theory of Computing Washington, DC, USA, ACM, 324–333.

[63] Hochbaum D. S., Shmoys D. B. (1986), A unified approach to approximation algorithms for bottleneck problems, Journal of the ACM, 33(3), pp. 533–550.

[64] Feder T., Greene D. H. (1988), Optimal algorithms for approximate clustering, In: Simon J ed. Proceeding of 20th Annual ACM Symposium Theory Computing, Chicago, Illinois, USA. ACM, pp. 434–444.

[65] Vaidya P. M. (1986), An optimal algorithm for the all-nearest-neighbors problem, In: Proceeding 27th IEEE FOCS Toronto, Canada, IEEE Computer Society, pp. 117–122.

[66] Megiddo N., Tamir A., Zemel E. *et al*. (1982), On the complexity of locating linear facilities in the plane, Operations Research Letters, 1, 194–197.

[67] Plesnik J. (1982), Complexity of decomposing graphs into factors with given diameters or radii, Mathematica Slovaca, 32(4), 379–388.

[68] Megiddo N., Tamir A. (1983), A new results on the complexity of p-center problems, SIAM Journal of Computing, 12(4), 751–758.

[69] Charikar M., Chekuri C., Feder T. *et al*. (1997), Incremental clustering and dynamic information retrieval, In: Leighton F T, Shor P W eds. Proceeding of 29th Annual ACM Symposium on Theory of Computing El Paso, Texas, USA, ACM, pp. 624–635.

[70] Agarwal P. K., Proeopiuc C. M. (1998), Exact and approximate algorithms for clustering, In: Karloff H J ed. Proceeding of 9th ACM-SIAM Symposium on Discrete Algorithms, San Francisco, California ACM/SIAM, pp. 658–667.

[71] Agarwal P. K., Proeopiuc C. M. (2000), Approximation algorithms for projective clustering, In: Shmoys D B ed. Proceeding of 11th ACM-SIAM Symposium on Discrete Algorithms, San Francisco, CA, USA, pp. 538–547.

[72] Bădoiu M., Har-Peled S., Indyk P. (2002), Approximate clustering via core-sets. In: Reif J H ed. Proceeding of 34th Annual ACM Symposium, Theory of Computing, Montréal, Québec, Canada. ACM, pp. 250–257.

[73] Khuller S., Moss A., Naor J. (1999), The budgeted maximum coverage problem. Information Processing Letters, 70, 39-45.

[74] Jain K., Vazirani V. (2001), Approximation algorithms for metric facility location and *k*-median problems using the primal–dual scheme and Lagrangian relaxation, Journal of the ACM 2001, 48, 274–296.

[75] Charikar M., Guha S. (1999), Improved combinatorial algorithms for the facility location and *k*-median problems, In: Proceedings of the 40th Annual Symposium on Foundations of Computer Science, New York, pp. 378–388.

[76] Hochbaum D. S., Maas W., (1985), Approximation schemes for covering and packing problems in image processing and VLSI, Journal of the ACM, 32, pp. 130–136.

[77] Glasser C., Reith S., Vollmer H. (2005), The complexity of base station positioning in cellular networks, Discrete Applied Mathematics, 148(1), 1–12.

[78] Erlebach T., Jansen K., Seidel E. (2001), Polynomial-time approximation schemes for geometric graphs, In: Kosaraju S R ed. Proceeding of the 12th Annual Symposium on Discrete Algorithms, Washington, DC, USA, ACM, pp. 671-679.

[79] Bilo V., Caragiannis I., Kaklamanis C *et al*. (2005), Geometric clustering to minimize the sum of cluster sizes, In: Montanari U, Rolim J D P, Welzl E eds. Proceedings of the European Symposium on Algorithms, LNCS, 3669, pp. 460–471.

[80] Gibson M., Kanade G., Krohn E. *et al*. (2008), On clustering to minimize the sum of radii, In: Teng S ed. Proceeding of the 19th Annual Symposium on Discrete Algorithms San Francisco, California, USA, SIAM, pp. 819–825.

[81] Arora S. (1996), Polynomial-time approximation schemes for Euclidean TSP and other geometric problems, In: Proceedings of the IEEE Symposium on Foundations of Computer Science, Burlington, Vermont, USA, IEEE Computer Society, 1996. pp. 2–12.

[82] Mitchell J. S. B. (1999), Guillotine subdivisions approximate polygonal subdivisions: a simple polynomial-time approximation scheme for geometric TSP, k-MST, and related problems, SIAM Journal of Computing, 1999, 28, 1298–1309.

[83] Bartal Y. (1996), Probabilistic approximations of metric spaces and its

algorithmic applications, In: Proceedings of the IEEE Symposium on the Foundations of Computer Science, Burlington, Vermont, USA, IEEE Computer Society, 1996, pp. 184–193.

[84] Fakcharoenphol J., Rao S., Talwar K. (2003), A tight bound on approximating arbitrary metrics by tree metrics, In: Larmore L L, Goemans M X ed. Proceedings of the ACM Symposium on the Theory of Computing, San Diego, CA, USA. ACM, pp. 448–455.

[85] Alt H., Arkin E., Bronnimann H *et al*. (2006), Mininum-cost coverage of points by disks, In: Amenta N, Cheong O ed. Proceedings of the Annual Symposium on Computational Geometry, Sedona, Arizona, USA, ACM, pp. 449–458.

[86] Li W. L., Zhang P., Zhu D. M. (2008), On Constrained facility location problems, Journal of Computer Science and Technology, 23(5), 740–748.

[87] Qian J., Wang C. A. (2006), How much precision is needed to compare two sums of square roots of integers?, Information Processing Letters, 100, 194–198.

[88] Demaine E. D., Mitchell J. S. B., O'Rourke J , The open problems project, Problem 33: Sum of square roots, http://maven.smith.edu/~orourke/TOPP/P33.html.

[89] Gibson M., Kanade G., Krohn G. *et al*. (2010), On metric clustering to minimize the sum of radii, Algorithmica, 57, 484–498.

[90] Gibson, M., Kanade, G., Krohn, *et al*. (2012), On clustering to minimize the sum of radii, SIAM Journal on Computing, 41(1), 47–60.

[91] Proietti G., Widmayer P. (2005), Partitioning the nodes of a graph to minimize the sum of subgraph radii, In: Deng X, Du D ed. Proceedings of the International Symposium on Algorithms and Computation (ISAAC), Sanya, Hainan, China, Springer, pp. 578–587.

[92] Behsaz B., Salavatipour M. R. (2012), On Minimum Sum of Radii and Diameters Clustering, In: Fomin F V, Kaski P ed. Proceeding of 13th Scandinavian Symposium and Workshops, Helsinki, Finland, Algorithm Theory – SWAT 2012, Lecture Notes in Computer Science, Springer, 7357, pp. 71–82.

[93] Jain R., Chaudhari N. S. (2012), A new 3-clustering algorithm for

minimum sum of diameter using bit representation. In: Proceeding of 7th IEEE Conference International Conference Industrial Electronics and Applications (ICIEA) Singapore, pp. 2004–2009.

[94] Ding C., Xiaofeng H., Hongyuan Z., Ming G., Simon H. (2001), A Min-Max Cut Algorithm for Graph Partitioning and Data Clustering, In: Proceedings of IEEE International Conference Data Mining, pp. 107–114.

[95] Y. G. Saab (2004), An Effective Multilevel Algorithm for Bisecting Graphs and Hypergraphs, IEEE Transaction, 641–652.

[96] Graham R.L., Knuth D.E., Patashnik O. (1988), Concrete Mathematics-A Foundation for Computer Science, 2nd Edition, Pearson Education.

[97] Van-Lint J.H., Wilson R.M. (2001), A course in Combinatorics, 2nd Edition, Cambridge University Press, pp. 119–128.

[98] Dagan I, Lee L., Pereira F. C. N. (1999), Similarity Based Model Of Word Co-Occurrence Probabilities, Machine Learning -Special issue on natural language learning archive, Kluwer Academic Publishers Hingham, MA, USA, 34, pp. 43–69.

[99] D. Bollegala, Y. Matsuo, M. Ishizuka (2011), A Web Search Engine-Based Approach to Measure Semantic Similarity between Words, IEEE Transactions On Knowledge And Data Engineering, 23(7), pp. 977-990.

[100] Rijsbergen C. J. V. (1979), Information Retrieval, Butterworth-Heinemann; 2nd edition.

[101] Y. Karov, S. Edelman (1996), Learning similarity-based word sense disambiguation from sparse data, In: Proceedings of the Fourth Workshop on Very Large Corpora, pp. 1–18.

[102] P. F. Brown, V. J. DellaPietra, P. V. deSouza, J. C. Lai, R. L. Mercer (1992), Class-based *n*-gram models of natural language, Computational Linguistics, 18(4), pp. 467-479.

[103] F. C. N. Pereira, N. Tishby, L. Lee, (1993), Distributional clustering of English words, In 31st Annual Meeting of the Association for Computational Linguistics, Somerset, New Jersey, pp. 183-190.

[104] G. Maltese, F. Mancini (1992), An Automatic Technique to Include Grammatical and Morphological Information in a Trigram- Based

Statistical Language Model, In: Proc. ICASSP, San Francisco, CA, pp. 157-160.

[105] Dagan S., Marcus S., Markovitch S. (1993), Contextual word similarity and estimation from sparse data, In 31st Annual Meeting of the ACL, Somerset, New Jersey, pp. 164–171.

[106] Dagan I., Marcus S., Markovitch S. (1995), Contextual word similarity and estimation from sparse data, Computer Speech and Language, 9, 123–152.

[107] Jardino M., Adda G. (1993), Automatic Word Classification, Using Simulated Annealing, In: Proc.1993 ICASSP, Minneapolis, MN,1993, pp. 41–44.

[108] Tamoto M., Kawabata T. (1995), Clustering Word Category Based on Binomial Posteriori Co-Occurrence Distribution, In: Proc. 1995 ICASSP, Detroit, MI, pp. 165–168.

[109] Magnus S., Cöster R. (2004), Using bag-of-concepts to improve the performance of support vector machines in text categorization, In: Proceedings of the 20th International conference on Computational Linguistics. Association for Computational Linguistics.

[110] Y. Liu, X. Wang, B. Liu, (2004), A Feature Selection Algorithm for Document Clustering based On Word Co-Occurrence Frequency, In: Proceedings of the Third International Conference on Machine Learning and Cybernetics, Shanghai, pp. 2963–2968.

[111] Zhou W., Du Y., Wang H., Lv X. (2011), Automatic identification of hierarchical relationship between words based on clustering, In International Conference on Transportation, Mechanical, and Electrical Engineering (TMEE), 2011, pp. 1585–1588.

[112] Marvit D., Jain J., Stergiou S., Gilman A., Adler B.T. (2009), Identifying Clusters of Words According to Word Affinities, Google Patents, EP2045736A1.