A UNIFIED APPROACH FOR CENTER-BASED CLUSTERING PROBLEMS ON NETWORKS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

DERYA İPEK EROĞLU

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
INDUSTRIAL ENGINEERING

JANUARY 2019

Approval of the thesis:

A UNIFIED APPROACH FOR CENTER-BASED CLUSTERING PROBLEMS ON NETWORKS

submitted by **DERYA İPEK EROĞLU** in partial fulfillment of the requirements for the degree of **Master of Science in Industrial Engineering Department, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences** ________________

Prof. Dr. Yaşar Yasemin Serin
Head of Department, **Industrial Engineering** ________________

Assoc. Prof. Dr. Cem İyigün
Supervisor, **Industrial Engineering, METU** ________________

Examining Committee Members:

Prof. Dr. Nur Evin Özdemirel
Industrial Engineering, METU ________________

Assoc. Prof. Dr. Cem İyigün
Industrial Engineering, METU ________________

Assoc. Prof. Dr. Pelin Bayındır
Industrial Engineering, METU ________________

Date:

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname: Derya İpek Eroğlu

Signature :

ABSTRACT

A UNIFIED APPROACH FOR CENTER-BASED CLUSTERING PROBLEMS ON NETWORKS

Eroğlu, Derya İpek
M.S., Department of Industrial Engineering
Supervisor: Assoc. Prof. Dr. Cem İyigün

January 2019, 133 pages

In this thesis, Center-Based Clustering Problems on Networks are studied. Four different problems are considered differing in the assignment scheme of the data points and the objective function. Two different assignment schemes are considered, hard assignment and soft assignment. In hard assignment, data points (vertices) are strictly assigned to one cluster, while in soft assignment, vertices are assigned to the multiple clusters with a membership probability. Objective function of a clustering problem could be categorized as minimizing sum of *distances* or sum of *squared distances* between the vertices and the centers of clusters they are assigned to.

In this study, cluster centers are not restricted to vertices. They are allowed to be located on vertices or anywhere on the edges. The problems that are studied are analyzed in terms of properties of the cluster centers, and theoretical results are derived. Benefiting from these properties, a unified solution framework is developed which is named Hybrid Genetic Algorithm (HGA), a genetic algorithm with a Local Search operation which uses the theoretical results obtained about the cluster centers. Two versions of HGA, namely Node Based HGA (HGA-N) and Edge Based HGA (HGA-

E) are developed by modifying HGA considering the derived properties. To test the performance of the proposed algorithms, numerical experiments are conducted on clustering of datasets from the literature and the simulated ones. Results are compared with the optimal or best solutions reported in the literature (if available). The proposed algorithms are also compared with the well-known heuristics used for the planar clustering problems. These heuristics are modified for the network problems. Computational results show that the proposed approach performs well in all clustering problems studied.

Keywords: Clustering on Network, Genetic Algorithm, Hard Assignment, Soft Assignment, Local Search

ÖZ

AĞLARDA MERKEZE DAYALI KÜMELEME PROBLEMLERİ İÇİN TÜMLEŞİK BİR YAKLAŞIM

Eroğlu, Derya İpek
Yüksek Lisans, Endüstri Mühendisliği Bölümü
Tez Yöneticisi: Doç. Dr. Cem İyigün

Ocak 2019 , 133 sayfa

Bu çalışmada, Ağlarda Merkeze Dayalı Kümeleme Problemleri üzerine çalışılmıştır. Noktaların küme merkezlerine atanma tipi ve ele alınan amaç fonksiyonu bakımından değişiklik gösteren dört farklı probleme odaklanılmıştır. Atama tipleri, katı atama ve yumuşak atama olarak iki sınıfa ayrılabilir. Katı atamada, veri noktaları (düğümler) bir kümeye katı olarak atanırken, yumuşak atamada, düğümler birden fazla kümeye üyelik fonksiyonu ile atanır. Çalışılan kümeleme problemlerinin amaç fonksiyonları, düğümler ile atandıkları merkezler arasındaki *uzaklıkların toplamını* enazlayan veya *uzaklıkların karesel toplamını* enazlayan fonksiyonlar olarak kategorize edilebilir.

Bu çalışmada, küme merkezleri düğümlerle kısıtlanmamış, küme merkezlerinin ağ üzerinde herhangi bir yerde olmasına izin verilmiştir. Çalışılan problemler, küme merkezlerinin davranışı ve amaç fonksiyonu bakımından incelenmiş ve birtakım teorik sonuçlar gösterilmiştir. Bu sonuçlardan faydalanılarak, Hibrit Genetik Algoritma (HGA) adını verdiğimiz, içinde Yerel Arama operatörü bulunan bir genetik algoritma olan, tümleşik bir çözüm yaklaşımı geliştirilmiştir. Elde edilen teorik sonuçlar kullanılarak, HGA yaklaşımının Düğüme Dayalı (HGA-N) ve Kenara Dayalı (HGA-V) olmak üzere iki tipi geliştirilmiştir. Bu algoritmaların performansını test edebilmek için, literatürden olan ve tarafımızca üretilen veri setleri kullanılmıştır. Sonuçlar, literatürde en iyi olarak verilmiş olan çözüm değerleri ile karşılaştırılmıştır (verildiği durumlarda). Önerilen algoritmalar, literatürde düzlem problemleri için bilinirliği olan sezgisel yaklaşımların ağ için modifiye edilmiş versiyonları ile karşılaştırılmıştır. Nümerik çalışmalar, önerilen yaklaşımın, çalışma kapsamında olan kümeleme problemleri için iyi bir performans sergilediğini göstermektedir.

Anahtar Kelimeler: Ağlarda Kümeleme Problemi, Genetik Algoritma, Katı Atama, Yumuşak Atama, Yerel Arama

To Chance...

ACKNOWLEDGMENTS

First of all, I cannot thank enough to Assoc. Prof. Dr. Cem İyigün, who taught me a lot besides how to do research. I learned a lot from this research experience I had with him. He always trusted me a lot, and I did my best to deserve his trust.

Second, I gratefully acknowledge the examining committee members Prof. Dr. Nur Evin Özdemirel, Assoc. Prof. Dr. Pelin Bayındır, Assoc. Prof. Dr. Alp Ertem and Assist. Prof. Dr. Bahar Çavdar for reviewing this work and providing invaluable feedback.

I would like to thank Duygu Pamukçu, who has been my best friend and officemate in both Turkey and United States. We started this journey of academia together, and we will be seatmates for a long time – maybe until retirement, who knows! I would like to thank Derya Dinler for her endless help and support. I learned so much from her experiences, and hopefully, I will keep learning from her. I also would like to thank Nick Brown for being supportive, encouraging and positive. He definitely made things easier in the last stage of my Master's study.

I could never thank enough to my mother Figen Yılmaz. I am very lucky to have such a strong and giving woman in my life. We have been through so much together, and I know that despite long distances, we will always be there for each other.

I would like to thank to my father Fatih Eroğlu for his endless support. The greatest thing I have ever learned from him is how to be passionate about the things I want to do in my life. I know that he will always be there for me, to listen Pink Floyd and Metallica songs together.

Lastly, I would like to thank all my colleagues for their support. Without them, my master's and assistantship experience would be less enjoyable. Also, I would like to thank all of the professors in METU IE for their contributions.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

CHAPTER 1

INTRODUCTION

Clustering is an unsupervised learning method that aims grouping data points that are similar, and separating data points that are dissimilar according to defined distance metric or similarity measure [2]. As stated in [3], clustering is useful for exploring the internal data structure. Clustering is a widely studied problem in the literature. In general, clustering approaches could be classified as

- Partitional (Center-based) clustering,
- Hierarchical clustering.

Partitional Clustering aims to form subsets of data points which are similar, while Hierarchical Clustering focuses on forming clusters with a hierarchical cluster structure.

Most of the studies regarding clustering have been performed in *continuous space*. In other words, data points with *m* attributes define an *m*-dimensional space, and cluster centers could be located on anywhere in the defined space. Some of the studies have been performed on a dataset which has *network* structure. Lately, graph clustering has become a popular topic and various algorithms have been proposed [4]. However, compared to the studies performed in continuous space, there are fewer studies performed on networks. Still, in the field of Location Theory, network location problems are widely studied in the literature. Spotting the similarities between particular types of network location problems and clustering problems on networks, some of the studies related to network location problems are utilized.

In this study, we examine *Partitional Clustering Problems on Networks* in which the number of clusters are known *a priori*. We focus on clustering problems on networks called *Euclidean Graph*. In these networks, the weight of an edge is Euclidean distance between the end vertices. These networks satisfy metric properties such as symmetry and triangle inequality. We deal with 4 problems which are listed as following.

- **P-Median Problem**: Hard Clustering with the objective of minimizing the sum of distances.
- **Sum of Squares Clustering Problem**: Hard Clustering with the objective of minimizing the sum of squared distances.
- **Probabilistic Distance Clustering (PD-Clustering) Problem**: Soft Clustering with the objective of minimizing the sum of distances.
- **Fuzzy Clustering Problem**: Soft Clustering with the objective of minimizing the sum of squared distances.

An illustration for the assignment types in clustering problems on networks is given in Figure 1.1. Here, two clusters are formed with a network of 30 vertices. In both cases, cluster centers are located on vertices 13 and 26. On the left network, data points are strictly assigned to their closest centers where it is called *hard assignment*. Each vertex belongs to one center where it leads to partitioning the network and creates two *disjoint* clusters. Clusters are shown with different shades of grey in the figure. On the right network, each vertex is assigned to both centers with a membership value where it is called *soft assignment*. Therefore, vertices are colored according to assignment or membership values.

In this study, we work on two soft clustering problems on networks and propose an approach for solving the problems. This approach can also be used for hard clustering that were already studied. With this way, we aim to bring a unified perspective to the partitional clustering problems on networks. By using the developed approach, we derive theoretical properties of the optimal solutions of the problems we focus. In this thesis, different than the general implementation, cluster centers are not restricted to the vertices on the network.They can also be located on the edges. We prove that for PD-Clustering Problem on networks, even when the cluster centers are allowed to be anywhere on the network, they will always be on vertices, while cluster

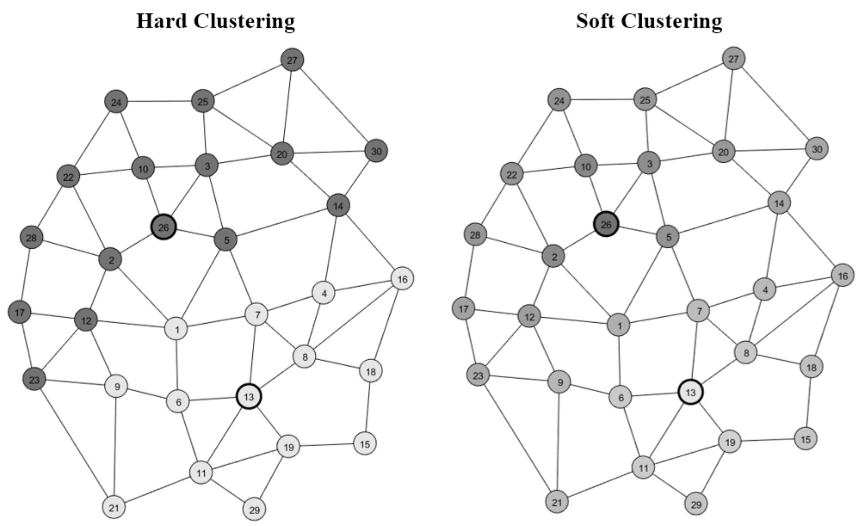

Figure 1.1: An illustration of hard clustering and soft clustering

centers could be anywhere in Fuzzy Clustering Problem. By analyzing properties of the objective functions of the problems, we derived some properties. As a solution approach, deriving these properties, we develop a Local Search (LS) procedure and embed it in a Genetic Algorithm (GA). This proposed approach is called as Hybrid Genetic Algorithm (HGA). HGA solves all the problems on hand. We developed two HGA versions, namely HGA-N and HGA-E, to solve these problems. We analyzed performances of the algorithms through computations on a data source from the literature and two different simulated datasets. Analysis show that both HGA-N and HGA-E algorithms have promising performance as far as solution quality is considered.

The organization of this thesis is as follows. In Chapter 2, literature review will be shared. In Chapter 3, notation that is used and definitions of some of the concepts will be provided. In Chapter 4, the problems listed previously will be defined and structural properties will be investigated. In Chapters 5 and 6, HGA approaches will be described and computational results will be provided. Lastly, in Chapter 7, conclusions of the study and future research directions will be discussed.

CHAPTER 2

BACKGROUND ON CLUSTERING AND LITERATURE REVIEW

Clustering problems are widely studied in the literature and state-of-art algorithms have been developed for the different versions of the problem. Similarly, Facility Location Problem, some variants of which could be treated as a clustering problem as well, has also been widely studied. In this chapter, benchmark studies in the literature regarding clustering problems are discussed. Although clustering on networks have been studied less than that on plane, it is a fairly popular area of study by researchers from various disciplines.

Partitional Clustering problems (regardless of their defined space) could be classified according to the assignment type:

- **Hard Assignment**: Data points are strictly assigned to the closest cluster center.
- **Soft Assignment**: Data points are assigned to centers with a membership value. This problem stems from Fuzzy Set Theory, in which items in a set belongs to the set with a probability.

Evaluation of the formed clusters is a requirement. In the studies where the number of clusters is given, commonly used objective functions in order to perform evaluation could be listed as following:

- **Sum of Distances:** Clustering problem is solved with the objective of minimizing the sum of distances between vertices and their centers.
- **Sum of Squared Distances:** Clustering problem is solved with the objective of minimizing the sum of squared distances between vertices and their centers.

Since clustering is a widely studied problem, a massive number of studies could be found in the literature, particularly for the studies performed on the plane. In this chapter, only the most well-known studies and the studies relevant to the problem of interest will be covered. In §2.1, clustering problems and its fundamental properties will be introduced to reader referencing the relevant literature. In §2.2, clustering problems defined on the plane will be discussed. Then, clustering on networks will be covered. Lastly, metaheuristic solution approaches proposed for the related problems in the literature will be discussed.

2.1 Clustering Problems

Data analysis techniques could be categorized as exploratory and confirmatory [5]. Exploratory data analysis aim to extract valuable information hidden in the data, while confirmatory data analysis aim to find properties in the data that could support a hypothesis or theory that is previously built. Regardless of the type of data analysis to be conducted, grouping similar objects or learning from labels of the data is important. At these point, learning algorithms, which could be dichotomized as supervised and unsupervised, take the stage. Supervised learning algorithms are basically classification algorithms that are trained to predict labels of the data based on the features on hand. Unsupervised algorithms deal with features of data and tries to find underlying structural properties in features that reveal similarities between data points. Clustering is an unsupervised learning problem that aims to group similar data points. With the help of Clustering, summarizing data that has high number of points and high number of features become easier. Since only the important characteristics could be observed with clustered data, summarizing the data becomes easier and more meaningful. This problem has applications in a wide range of disciplines, such as medicine, archaeology, biology, economy, market research, and linguistics [2].

Since clustering problems are studied by a wide range of disciplines, there are a wide range of approaches. In the literature reviews [6], [7] and [5], clustering approaches were classified. Despite not being exactly the same, these classifications are similar to each other. The classification scheme could be summarized as following. In the following subsections, these categories will be covered in further detail.

- Hierarchical Clustering
- Partitional Clustering
  - Clustering on Plane
  - Clustering on Networks

2.1.1 Hierarchical Clustering

In this approach, clusters are formed in a way that they have a hierarchy. For example, a data point is a member of a cluster, and that cluster is a member of another cluster – there is a hierarchical structure. And there is one cluster on very top-level that contains all the data points. Algorithms for hierarchical clustering could be categorized as *Agglomerative* and *Divisive*. Agglomerative algorithms start with creating individual clusters for each data point, and combine clusters in order to minimize a performance measure called *linkage*. Two traditional algorithms are single linkage and complete linkage algorithms. In single linkage, two clusters the closest members of which have the smallest distance are combined to form a cluster. In complete linkage, two clusters the farthest members of which have the smallest distance are combined to form a cluster [8]. Unlike Agglomerative algorithms, Divisive algorithms start with one cluster that contains all the data points, then split this cluster until obtaining individual clusters for each data point. Divisive algorithms are computationally expensive, since they check all subset pairs for splitting. Therefore, Agglomerative clustering algorithms are used more [7]. Hierarchical Clustering is advantageous in that it works with various levels of granularity in the data [6]. One disadvantage of Hierarchical Clustering could be that it is computationally expensive compared to traditional Partitional Clustering algorithms.

2.1.2 Partitional Clustering

This approach aims to partition the data into several subsets according to similarity. Since it is impossible to check all the possible partitions, greedy approaches were proposed such as K-means [9] and K-medoids [10] algorithms. These algorithms work in an iterative manner and try to minimize an objective function. This objective function usually depends on the distance between data points and the closest cluster center. Clusters are mutually exclusive since hard assignment is made, that is, each data point is assigned to one cluster. Another approach could be soft assignment in which data points are assigned to all clusters with a probability. Regardless of the assignment scheme, these algorithms obtain clusters that are convex in shape. To obtain nonconvex clusters, approaches that consider cluster density could be implemented, aka Density-Based Clustering [6]. An algorithm example could be DBSCAN. In this approach, dense regions are found by scanning neighborhood of points, and these dense regions are connected to each other according to their connectivity.

2.2 Background on Partitional Clustering Problems and Solution Approaches

In this section, partitional clustering solution approaches will be discussed. In subsection 2.2.1, solution approaches for clustering algorithms solution space of which is defined on planes will be discussed. Then, clustering algorithms on networks will be described in subsection 2.2.2. In 2.2.3, metaheuristic approaches proposed for partitional clustering problems will be mentioned.

2.2.1 Clustering on Plane

There are four different clustering problems relevant to this study. These problems differ in assignment types and objective functions, as represented in Figure 2.1. In this subsection, these problems and their most known solution approaches will be discussed in detail.

For the clustering problem with hard assignment and minimization of the sum of squared distance between each point and center of its assigned cluster, the objective function is given in (2.1), where I is the set of data points, K is the set of clusters, x_i is coordinates of a data point i, c_k is coordinates of centroid of cluster k, and $d(x_i, c_k)$ is the distance between point i and cluster center k (The distance is typically Euclidean). K-means algorithm was proposed by Hartigan in 1979 [9]. A brief pseudocode of the algorithm has been given in Algorithm 1. The algorithm proceeds in an iterative way.

Assignment Type

Objective Function		Hard Assignment	Soft Assignment
	Sum of distance	PAM (K-Medoids)	PD-Clustering
	Sum of squared distance	K-Means	Fuzzy C-Means

Figure 2.1: A classification for partitioning-based clustering algorithms on plane

The procedure begins with generating initial centroid locations. This generation is usually made randomly. Then, allocations of points to clusters are calculated. With these allocations, new centroid coordinates are calculated. With the obtained new coordinates, allocations are calculated, and these location-allocation steps are repeated until allocations do not change, i.e., solution converges.

$$minimize \sum_{i \in I} \min_{k \in K} \{d(x_i, c_k)^2\} \tag{2.1}$$

Algorithm 1 K-means Algorithm

Input: *Dataset and k (the number of clusters)*

Output: *Centroid coordinates and Allocations*

Generate initial coordinates for *k* centroids

Compute assignment of each data point to clusters

while Assignments of data points change **do**

  Compute new centroid coordinates for each cluster

  Compute new allocation of each data point to clusters

end while

$$minimize \sum_{i \in I} \min_{k \in K} \{d(x_i, c_k)\} \tag{2.2}$$

As an hard assignment problem that aims to minimize the sum of distances between each point and center of its assigned cluster, K-medoids problem focuses on selecting centers of the clusters among data points. Since this problem employs the sum of distances as the objective, (see (2.2)), it is less sensitive to noise and outliers than K-means algorithm [10]. One of the proposed algorithms is called PAM (Partitioning Around Medoids) [10], pseudocode of which is given in Algorithm 2. This algorithm consists of two steps: BUILD and SWAP. BUILD step is generating an initial solution. After build step, for each non-medoid data point, change in the objective function when selecting that point as medoid of its cluster and unselecting the current medoid is calculated and the data point that decreases the objective function value the most is selected as medoid. This procedure is called SWAP. This operation is repeated until the objective function value does not improve. This algorithm could be classified as a greedy algorithm. Still, it could be said that PAM is a benchmark algorithm.

Algorithm 2 PAM Algorithm

```
Input: Dataset and k (the number of clusters)
Output: Medoid coordinates and Allocations
Select k initial medoids among the data points
Compute allocation of each data point to clusters
while The objective function improves do
    Perform the swap that decreases objective function the most for each medoid
end while
```

Soft assignment concept was inspired from Fuzzy Set Theory. In Fuzzy Set Theory, an object belongs to a set with a membership value, which is between $[0, 1]$. Dividing the data into strict groups may not match real structure of the data [11]. The first approach inspired from this idea was developed by Bezdek, which is the *Fuzzy C-Means* (FCM) algorithm [12]. The objective function of this algorithm is given in (2.3), where p_{ik} is the membership value of point x_i to center c_k, and m is the fuzziness constant. As m increases, the sets get fuzzier (memberships converge to $\frac{1}{|K|}$). Given the center locations, membership values are calculated as in (2.4). Centers of the clusters are calculated as (2.5). Pseudocode is given in Algorithm 3. The algorithm terminates when objective function value converges.

$$minimize \sum_{i \in I} \sum_{k \in K} p_{ik}^m d(x_i, c_k)^2 \tag{2.3}$$

$$p_{ik} = \frac{1}{\sum_{j \in K} \frac{d(x_i, c_k)}{d(x_i, c_j)}^{\frac{2}{m-1}}} \tag{2.4}$$

$$c_k = \frac{\sum_{i \in I} p_{ik}^m x_i}{\sum_{i \in I} p_{ik}} \tag{2.5}$$

Algorithm 3 *FCM* and *PD-Clustering* Algorithms

Input: *Dataset and k (the number of clusters)*

Output: *Centroid coordinates and Membership values*

Generate initial coordinates for *k* centroids

Compute memberships of each data point to each cluster

while Stopping condition is not met **do**

  Given the memberships, compute new centroid coordinates for each cluster

  Given the centers, compute new memberships of each data point to each cluster

end while

The algorithm that performs soft assignment with the objective (2.6) has been developed by Iyigun and Ben-Israel in [13], named as *Probabilistic Distance Clustering* (*PD-Clustering*). This algorithm proceeds in an iterative way as well, pseudocode is given in Algorithm 3. Membership values are calculated as in (2.7), and center coordinates are calculated as in (2.8) when distance is defined as Euclidean distance. The algorithm terminates when the center locations do not change anymore.

$$minimize \sum_{i \in I} \sum_{k \in K} p_{ik}^2 d(x_i, c_k) \tag{2.6}$$

$$p_{ik} = \frac{1}{\sum_{j \in K} \frac{d(x_i, c_k)}{d(x_i, c_j)}} \tag{2.7}$$

$$c_k = \frac{\sum_{i \in I} u_{ik} x_i}{\sum_{i \in I} u_{ik}}, \text{ where } u_{ik} = \frac{p_{ik}^2}{d(x_i, c_k)} \tag{2.8}$$

Besides these algorithms, Genetic Algorithm (GA) based approaches have been proposed for K-Means and K-Medoids problems. GAs are first proposed by Holland in [14], stating that if nature is simulated by computer systems, even the complicated problems could be solved.

In [15], a hybrid GA was proposed for K-Means problem. As chromosome representation, they store assignments of each data point. This algorithm does not have a traditional crossover operator. Instead, it performs one iteration of *K-Means* algorithm. The mutation operator changes assignment of a randomly selected data point with to a cluster with a probability inversely related to its distance. It is reported that Genetic K-means Algorithm outperforms *K-Means* algorithm in terms of solution quality since *K-Means* usually converges to local optimum –even when multistart is made.

For K-Medoids problem, another hybrid GA was proposed in [16]. In the proposed approach, number of clusters to be formed is assumed to be unknown. As chromosome representation, they store vertex objects that are medoids. This algorithm randomly decides number of clusters each individual has. There is a crossover operator that could be seen traditional, and a mutation operator changes a medoid randomly. Furthermore, the algorithm has a heuristic operator that is basically a local search procedure. It is reported that performance of the algorithm is promising when solution quality is compared to other well-known algorithms, such as *CLARA*, which was proposed in [10].

2.2.2 Clustering on Networks

Clustering on networks could be defined as finding sets of similar vertices if vertices are assumed to be data points. An extensive survey has been performed by Schaeffer [4]. Another survey has been conducted by Aggarwal and Wang in [17]. Most of the approaches covered in these surveys are not based on locating the center. They handle the problem as a graph clustering or graph partitioning problem.

One of the problems defined is Minimum-Cut Clustering (MCC) [18]. This problem is based on the Minimum Cut Problem in [19], which is a well-known one among network flows problems. In the Minimum Cut Problem, the aim is to obtain two clusters from the graph that minimizes distance within the partitions. In MCC, partitions found for k clusters while minimizing distance within the clusters. An exact solution approach has been developed that defines the problem as a Mixed Integer Programming model and solves it with decomposition and column generation.

Another approach has been developed by Lin and Kerninghan in [20]. The aim is to partition the vertices that minimizes the number of intercluster connections. In the proposed algorithm, pairwise exchange between clusters that decreases the objective the most is performed. With this way, pairwise optimality is aimed, which is a necessary condition for optimal partitions.

In [21], derivation of community structure of natural networks such as social networks and biological networks have been studied. The developed algorithm created a partition of a network by removing edges from the original graph, and the removed edges are the ones connecting different communities. In order to determine which edge to remove, they used a measure called *Edge Betweenness Centrality*. This is a measure for number of shortest paths passing through a certain edge. The edges that have high *Edge Betweenness Centrality* value is considered as *bridge*, and removed from the graph.

The approaches discussed so far in this subsection are three of the well-known algorithms for graph clustering. Their common feature is that they are not center-based, i.e., a cluster center concept is not utilized in these studies. There are two main problems that treat the clustering problem on networks in a center-based manner.

The first problem is known as P-Median Problem. This problem is widely studied, and there is a massive number of publications. The initial studies about this problem regarding the definition and theoretical studies are performed by Hakimi, see [22] and [23]. Objective function of the problem is given in (2.9), where $d(v_i, x_k)$ refers to the length of the shortest path between vertex i and cluster center k. The objective function is to minimize sum of distances of each vertex to their closest cluster center. In [22], Hakimi proved that there is an optimal solution in which cluster centers are located at vertices. This proof covers only the 1-Median case. Then, in [23], he proved that the proof is generalizable for P-Median Problem. Levy proved that these proofs are a consequence of concavity of the objective function in [24], which will be covered in Chapter 4 in detail.

$$minimize \sum_{i \in I} \min_{k \in K} \{d(v_i, x_k)\} \tag{2.9}$$

An extensive literature survey for P-Median Problem could be found in [25]. In 1986, a GA has been proposed by Hosage and Goodchild [26]. After that, in [27], a more efficient GA has been developed by Alp and Erkut. They reported that their solution quality on ORlib instances is promising.

Another problem defined for network clustering that is center-based is made by Carrizosa et. al. in [1]. The problem is defined as Sum of Squares Clustering, objective function of which is given in (2.10). The objective is to minimize sum of squared distance between each vertex and center of its cluster center. They found that the objective function is convex. Therefore, they report that there could be centers on edges of the graph which yield the optimal objective value.

$$minimize \sum_{i \in I} \min_{k \in K} \{d(v_i, x_k)\}^2 \tag{2.10}$$

As a solution approach, they proposed an implementation of Variable Neighborhood Search (VNS) heuristic. They perform search not only on vertices, but also on edges, and they report objective function values for ORlib instances for two cases: heuristic solution when centers are only located on vertices and heuristic solution when centers are located on vertices and edges. It was reported that there are edge solutions that have lower objective function value than vertex solutions.

2.2.3 Metaheuristic Solution Approaches

As discussed in previous sections, there are various solution approaches developed for P-Median Problem. Among them, two of the studies propose a Genetic Algorithm (GA) for the problem. The first one has been developed by Hosage and Goodoffspring in [26]. This algorithm utilizes *binary encoding* as chromosome representation, that is, they store a binary array representing whether a vertex has been selected as a cluster center or not. Since this encoding scheme leads to encountering infeasible solutions (solutions with number of clusters different than *p*), an operator that repairs or eliminates infeasible solutions is required. This tends to make the algorithm computationally inefficient.

In [27], Alp et.al. developed another genetic algorithm with a more efficient representation scheme. That is, for each cluster, index of the vertex that was selected as cluster center has been stored. With this way, feasibility of the solution has been guaranteed. They report that they outperform [26] with the new GA. However, this algorithm has inefficient operations as well. First, the population is not generated randomly. Therefore, they need to work with a large-size population to represent solution space. Second, as crossover operator, they take the union of two solutions and perform greedy deletion. For example, if there are p clusters and two completely different parents are taken, their union will give a solution with $2p$ clusters. By deleting the cluster centers that causes the lowest increase in the fitness function, they obtain a solution with p clusters. This procedure could become computationally expensive especially when the population size gets larger.

For Sum of Squares Clustering Problem, Carrizosa et.al. developed a Variable Neighborhood Search (VNS) based metaheuristic in [1]. In this metaheuristic, an initial solution is generated randomly. Then, it is passed to a procedure named *Net K-Means* that is basically a *K-Means* algorithm works on network. First, given the assignments, optimal location for each cluster center is found. Then, given the center locations, optimal assignments are found, and this procedure is repeated until the solution converges. In the local search procedure, they perform search not only on vertices, but also on edges. Fixing the assignments, the best location for each cluster center on network is found. Then, fixing the locations, assignments of vertices to clusters are changed. This procedure iterates until solution converges. After the solution converges, an operation is performed to generate a new initial solution for the *Net K-means* algorithm which is called *Shaking*. By means of *Shaking* procedure, the problem of sticking into local optimal is alleviated. The generated solution is random and it is selected within the defined neighborhood of that iteration. The algorithm iterates until no improvement could be made on the solution. This algorithm has an advantage that it performs search on the edges as well. However, this algorithm has two main downsides. First, since it performs edge search on the network, finding optimal location by searching all the edges may take a long time. Second, since *Net K-means* has been applied, by its nature, the assignments are fixed during the edge search. This may bring the user to a local optima or slow down the convergence.

Krishna and Narasimha Murty developed a Hybrid GA for the K-Means Clustering Problem in [15]. In this algorithm, *string-of-group-numbers* encoding was used as chromosome representation. For each individual, a matrix with the size of number of data points by number of clusters are stored. If an individual belongs to cluster k, k^{th} element of the array is 1 and others are 0. The initial population is randomly generated so that none of the clusters are empty. In generation replacement stage, *Roulette Wheel Selection* is used based on the fitness function, which is Sum of Squared Errors (here, error refers to distance between data point and centroid of its cluster). In mutation operation, a point has been assigned to another cluster with a probability increasing with distance to that cluster in a way that none of the clusters are empty. The algorithm does not have a conventional crossover operator. Instead, there is a *K-Means* operator which performs one iteration of *K-Means* algorithm to the individual. Due to existence of the *K-Means* operator, the algorithm is called *hybrid*. With the help of the *K-Means* operator, they report that the solution converges faster, and this could be seen as an upside of the approach. The downside of the approach could be the chromosome representation. Since chromosome representation is binary, empty clusters could be encountered, and this must be checked for all individuals. Since crossover does not exist, more than a GA, this algorithm could be seen as a randomized search algorithm.

In [16], a Hybrid GA has been proposed for K-Medoids Clustering. In this study, number of clusters are not given *a priori*, and finding optimal number of clusters is aimed as well. To do this, a suitable validity index is used as a fitness function. Therefore, they proposed to use Davies-Bouldin index. The chromosome representation is an array with the size of number of clusters for each individual. In the array, points selected as medoids are stored. Since the number of clusters are unknown, chromosome size could differ among individuals. Therefore, the operators must be specially designed regarding this condition. After selecting random parents, crossover operation is carried out. The crossover operation randomly produces two offsprings that is not identical to each other or their parents. Number of clusters these offsprings have could be different, but it should be in $[2, k_{max}]$, where k_{max} is a predefined parameter. Mutation operation is performed with a probability. In this operation, a medoid is replaced with a random point that does not exist in the population. Furthermore, in order to boost the convergence, they used a heuristic operator that aims to minimize sum of total distance between points and their assigned medoids. It is reported that heuristic operator has a significant effect on the convergence. One advantage of this approach is that number of clusters are optimized besides finding the medoids. Additionally, with heuristic operator, algorithm will definitely converge faster. However, the heuristic operator should be chosen carefully because there is a trade-off between solution quality and computational effort. Especially in large size populations, since heuristic operator is needed to be run much more, computational effort could be significantly high.

In the light of the discussions above, we can say that P-Median Problem is widely studied. Sum of Squares Clustering Problem on networks is recently studied. Different from the literature, we defined two new problems on networks, which are PD-Clustering and Fuzzy Clustering Problem. Inspiring from the Location Theory, we analyzed all of these four problems with a framework, and with the derived properties, we developed a solution approach that can solve all these four problems. In short, we not only study two new problems, but also implement a framework to bring out the similarities of the problems on hand.

CHAPTER 3

NOTATION AND FUNDAMENTALS

In this chapter, notation used throughout this thesis is described in §3.1. Then, in §3.2, definitions that have been made in the literature and used in this study are discussed. Furthermore, *Assignment Bottleneck Point* which is newly introduced with this thesis are discussed.

3.1 Notation and Fundamentals

The problems covered in this thesis have been defined on a graph $\mathbf{G} = (\mathbf{E}, \mathbf{V})$where $\mathbf{V}$ refers to the set of vertices and $\mathbf{E}$ refers to the set of edges. $\mathbf{G}$ is an undirected and connected graph, that is, all vertices are connected. Therefore, $|\mathbf{E}| \geq |\mathbf{V}| - 1$. In other words, for being connected, the number of edges must be at least the *number of vertices-1*. On graph $\mathbf{G}$, $d(v_i, x_k)$ is defined as length of the shortest path distance between vertex *i* and cluster center *k*. Since $\mathbf{G}$ is an undirected graph, $d(v_i, x_k)$ =$d(x_k, v_i)$. The distance on $\mathbf{G}$ has been defined as a metric distance such as Euclidean Distance. In other words, the distance is assumed to satisfy the metric properties. Different distance measures could be used in this framework. We assume Euclidean Distance, and we work on Euclidean Graphs. The notation used in this chapter is given in Table 3.1.

3.2 Definitions

In this section, *Arc Bottleneck Point*, *Assignment Bottleneck Point* and *Finite Dominating Set* will be defined. These definitions will be needed later in deriving theoreti-

Table 3.1: Table of Notations

$\mathbf{V}$	Set of vertices
$\mathbf{X}$	Set of cluster centers
n	Number of vertices $\mathbf{V}$
$\mathbf{I}$	Index set of vertices $I = \{1, 2, ..., n\}$
v_i	Vertex i, $i \in \mathbf{I}$
h_i	Weight of v_i ($h_i > 0\ \forall i \in \mathbf{I}$)
b_i	Arc bottleneck point of v_i on the edge (v_p, v_q)
a_i	Assignment bottleneck point of v_i
p	Number of clusters $\lvert\mathbf{X}\rvert$
x_k	Location of cluster center k
$d(v_i, x_k)$	Length of the shortest path from v_i to x_k
p_{ik}	Probability of assignment of v_i to cluster k

cal properties of the problems on hand.

3.2.1 Arc Bottleneck Point

Let v_i be an arbitrary vertex on $\mathbf{G}$, e_{pq} be an edge on $\mathbf{G}$ that connects vertices v_p and v_q, and l_e be length of the edge. For any point $x \in e_{pq}$, $d(v_i, x)$, length of the shortest path from v_i to x is calculated as

$$d(v_i, x) = \min\{d(v_i, v_p) + d(v_p, x), d(v_i, v_q) + d(v_q, x)\}. \tag{3.1}$$

This means the shortest path to x passes from either v_p or v_q. The distance function from v_i to x has been given in Figure 3.1. There exists a point, b_i, on the edge e_{pq} at which lengths of the shortest paths using vertices v_p and v_q are equal. It is the farthest point to vertex v_i on the edge e_{pq}. The point is calculated as

$$d(v_i, v_p) + d(v_p, b_i) = d(v_i, v_q) + d(v_q, b_i)$$

Let $d(v_q, b_i) = l_e - d(v_p, b_i)$. Then,

$$\begin{aligned} d(v_i, v_p) + d(v_p, b_i) =& d(v_i, v_q) + l_e - d(v_p, b_i) \\ d(v_p, b_i) =& \frac{1}{2}(d(v_i, v_q) + l_e - d(v_i, v_p)), \end{aligned} \tag{3.2}$$

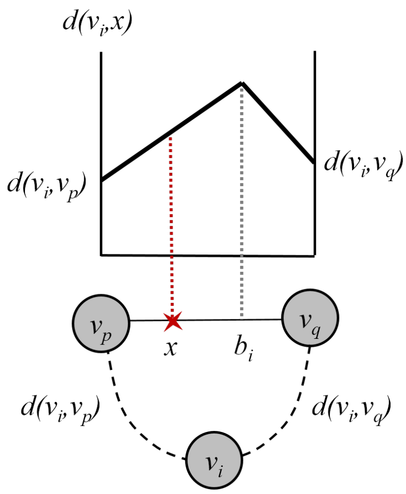

Figure 3.1: Distance function $d(v_i, x_)$ on the edge connecting v_p and v_q

where b_i is called as *arc bottleneck point* [28]. Before the study in [28], Hakimi used this concept in his proof in [22]. There are three cases regarding the value of $d(v_p, b_i)$:

Case 1 . If $d(v_p, b_i) \leq 0$, the shortest path from v_i to x always passes from v_q.

Case 2 . If $d(v_p, b_i) \geq l_e$, the shortest path from v_i to x always passes from v_p.

Case 3 . If $d(v_p, b_i) \in (0, l_e)$, the shortest path from v_i to x passes from:

- v_p if $d(v_p, x) <= d(v_p, b_i)$,
- v_q if $d(v_p, x) > d(v_p, b_i)$.

This distance function is linear or piecewise concave on the edge e_{pq}, and *arc bottleneck point* is the point where piecewise behavior occurs. In the case of more than one vertices, this behavior is valid. For example, in Figure 3.2, distance function as a summation of $d(v_i, x)$ and $d(v_j, x)$ is given. It could be observed that there are two bottleneck points this time. Since we consider two vertices in the function $d(v_i, x) + d(v_j, x)$, each bottleneck point corresponds to one of the vertices. Since end vertices of an edge will not make arc bottleneck points, number of arc bottleneck points on an edge is at most *n-2*.

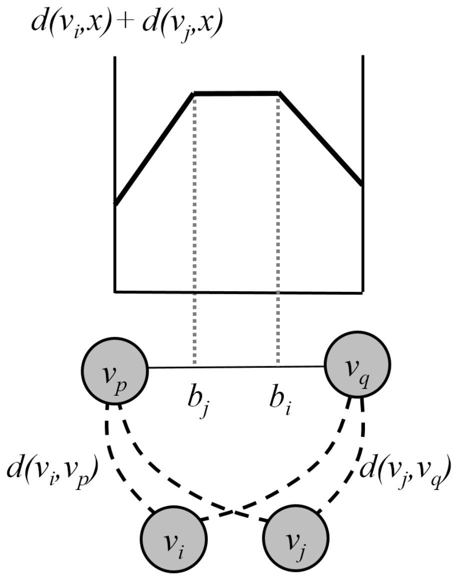

Figure 3.2: Distance function $d(v_i, x) + d(v_j, x)$ on the edge connecting v_p and v_q

3.2.2 Assignment Bottleneck Point

Arc bottleneck point does not consider assignment changes while changing the center locations. We introduce a new concept of bottleneck point that also accounts for assignment changes, and we call it as *assignment bottleneck point*.

Let v_i be an arbitrary vertex on $\mathbf{G}$, $x_k \in e, k = 1, ..., K$ be the closest center to v_i, e_{pq} be an edge on $\mathbf{G}$ that connects vertices v_p and v_q, and l_e be the length of the edge e_{pq}. If $x_l \in \mathbf{G}$ is the cluster center that is the second closest to v_i and all center locations except x_k are fixed, as x_k changes on e_{pq} and so $d(v_i, x_k)$ changes, the closest center to v_i may change. The point at which the closest center to v_i switches is called *Assignment Bottleneck Point*. Figure 3.3 shows an illustration of the assignment bottleneck point a_i. For example, let $d(v_i, x_k)$ =13, the shortest path to x_k passes from v_q, and $d(v_i, x_l)$ =15. If x_k is moved towards v_p, $d(v_i, x_k)$ increases (It is assumed that there is not an arc bottleneck point on e_{pq}.). When x_k is moved more than 2 units towards v_p, the closest center to v_i will be x_l since $d(v_i, x_k) \geq d(v_i, x_l)$. Therefore, the closest center to v_i is no longer x_k. Since the location of x_l is fixed, the distance of v_i to the closest center, $\min_{j \in K}\{d(v_i, x_j)\}$, remains constant as x_k continues moving towards v_p.

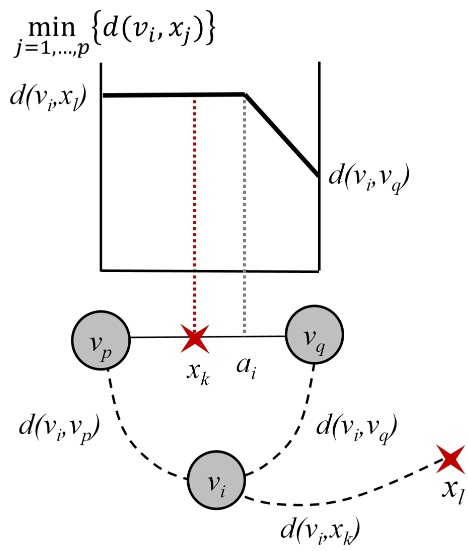

Figure 3.3: Assignment Bottleneck Point

Figure 3.4 illustrates the case under existence of an arc bottleneck point. As x_k is moved towards v_p and $d(v_i, x_k)$ increases, after the point a_i^q, x_l becomes the closest center. The black dashed line illustrates change in $d(v_i, x_k)$. When x_k reaches b_i, which is the arc bottleneck point, $d(v_i, x_k)$ takes its maximum value along e_{pq}. After b_i, value of $d(v_i, x_k)$ decreases. When $d(v_i, x_k) = d(v_i, x_l)$ (shown as a_i^p), x_k becomes the closest center to v_i again. As a result, we observe two assignment bottleneck points a_i^p and a_i^q on edge e_{pq}. Above, we have covered two cases of *assignment bottleneck point*. A variety of these situations could happen. Under the existence of *arc bottleneck point*, it is possible to have only one assignment bottleneck point – or no assignment bottleneck points. On a given edge, a vertex can create at most two assignment bottleneck points. For a given vertex to have two assignment bottleneck points on edge e_{pq}, it must have an arc bottleneck point as well. Hence, on each edge, there could be at most $2(n-2)+2$ assignment bottleneck points.

3.2.3 Finite Dominating Set (FDS)

Finite Dominating Set (FDS) was introduced by Hooker et. al. in [28]. Given a graph $\mathbf{G} = (\mathbf{E}, \mathbf{V})$, where $\mathbf{E}$ is set of edges and $\mathbf{V}$ is set of vertices, FDS is defined as a set of points on $\mathbf{G}$ where optimal solution of a problem must belong. For different network

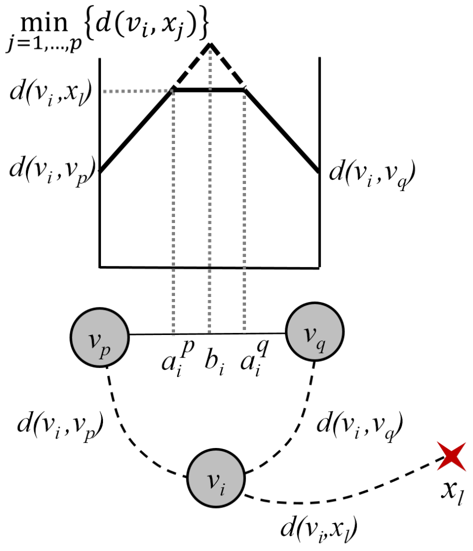

Figure 3.4: Assignment Bottleneck Point with Arc Bottleneck Point

location problems, FDS is defined differently depending on structural properties of the problem on hand. These properties mainly depend on convexity/concavity of objective function. Finding an FDS for a problem is clearly beneficial because it reduces the solution space and complexity. This study could be seen as a guideline for the researchers who work on a various range of Network Location Problems. p-median and SSC problems' structural differences defines different FDSs.

In 1964 and 1965, Hakimi proved that FDS of P-Median Problem is $\mathbf{V}$ ([22], [23]). In 1967, Levy proved that this is a consequence of objective function's being concave [24]. Based on this, in [28], it is suggested to check concavity of transportation cost to find FDS of any network location problem as a strategy.

In [1], Sum of Squares Clustering on Networks has been studied. They found that the objective function is a second-degree polynomial between consecutive arc bottleneck points. Therefore, they reported FDS of this problem as set of local minimum points on $\mathbf{G}$. Therefore, cluster center could be located on edges as well as vertices.

FDS is a fundamental concept of this study since we studied on finding structural properties like FDS of the problems on hand.

CHAPTER 4

THEORETICAL RESULTS FOR CLUSTERING

As stated previously, there are four different clustering problems that are considered in this study. These four problems differ in assignment schemes and objective functions they use. Inspired by approach used to obtain results for different network location problems in [28], we implemented a framework to these four problems. The main aim behind analyzing these problems is to find properties of the optimal solution of each problem.

This chapter is organized as follows. In §4.1, theoretical results for hard assignment problems will be discussed. Soft assignment problems that are newly defined on networks, and theoretical results derived for these problems will be discussed in §4.2.

4.1 Clustering Problems on Networks with Hard Assignment

In this section, two clustering problems that consider hard assignment will be discussed. In hard assignment case, each vertex is assigned to one cluster, center of which is the closest. There are two hard clustering problems that will be covered. First, P-Median Problem that has the objective function of minimizing the sum of distances between vertices and their centers will be discussed. Then, sum-of-squares clustering (SSC) problem that has the objective function of minimizing the sum of squared distances between vertices and their centers will be defined and analyzed.

4.1.1 P-Median Problem

The P-Median Problem is defined as

$$\textit{minimize } f(\mathbf{X}) = \sum_{i=1}^{n} h_i \min_{k=1,\dots,p} \{d(v_i, x_k)\}$$

$$\text{subject to}$$

$$x_k \in V \quad \forall\, k = 1, \dots, p,$$

where $\mathbf{X}$ is the vector of center locations and x_k is decision variable for location of cluster center k, and h_i is a nonnegative constant showing weight of vertex v_i. Since hard assignment is considered, each vertex is assigned to one of the centers. For each vertex, only the distance between the vertex and its closest center (multiplied by vertex weight) contributes to the objective function. In this subsection, first, P-Median Problem with a single cluster (1-Median Problem) will be analyzed. Then, P-Median Problem will be discussed.

1-Median Problem

Suppose we have only one cluster and one cluster center will be located on $\mathbf{G}$. In that case, the formulation is

$$\textit{minimize } f(x_c) = \sum_{i=1}^{n} h_i d(v_i, x_c) \tag{4.1}$$

$$\text{subject to}$$

$$x_c \in \mathbf{G}. \tag{4.2}$$

Let us first consider the line graph as in Figure 4.1 which has 4 vertices. Under the assumption that cluster center could be located anywhere on the graph, there is a center located between vertices 2 and 3. $d(v_1, v_2) = a$, $d(v_2, v_3) = l$ and $d(v_3, v_4) = b$ are given. Let y denote the distance of the center from vertex 2. Assuming that all vertex weights has the value of 1, the objective function to minimize total distance between vertices and the center is

$$f = (a + y) + y + (l - y) + (b + l - y) = a + 2l + b,$$

which is constant. This implies that any location of the center on $\mathbf{G}$ will lead to the same objective function value. So, the center could be located on vertices or edges.

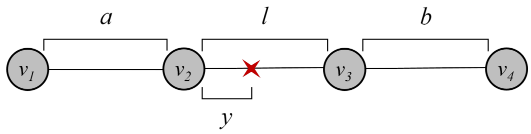

Figure 4.1: A Line Graph with 4 vertices

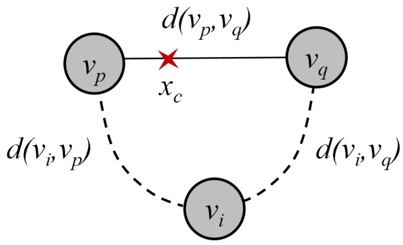

Figure 4.2: An illustration of a part of a graph $\mathbf{G}$

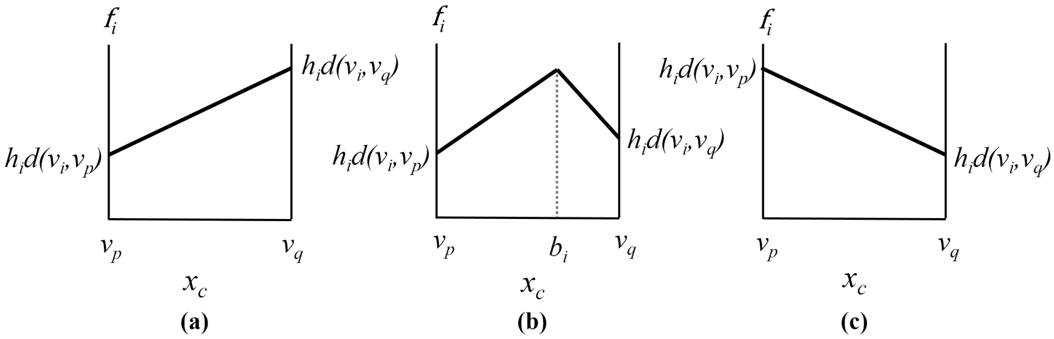

Figure 4.3: Objective function component for v_i (denoted as f_i) when x_c is moved along the edge (v_p, v_q)

A general network is illustrated in Figure 4.2 in which cluster center x_c is on an edge e_{pq} connecting (v_p, v_q). In the case of this network, we may observe three different patterns of objective function component of a vertex v_i to the (4.1) depending on location of x_c. The shortest path from v_i to x_c may pass from v_p or v_q regardless of the location of x_c, which also means that there is no arc bottleneck point. If there is an arc bottleneck point on e_{pq}, the shortest path to x_c will pass from v_p or v_q depending on the location of x_c on e_{pq}. These patterns are shown in Figure 4.3. In the Figure, (a) is the case when shortest path to x_c passes from v_p, and (c) is the case when shortest path to x_c passes from v_q. In both cases, there is no arc bottleneck point. In (b), if x_c is in the interval $[v_p, b_i]$, shortest path to x_c passes from v_p; otherwise, shortest path to x_c passes from v_q. The reason of this behavior is the bottleneck point b_i observed. In all of these cases, it could be observed that the objective function is linear or piecewise concave.

Based on these patterns discussed above, in [22], Hakimi proved that $\mathbf{V}$ contains optimal solution. In [24], Levy proved that this proof is a result of concavity of the objective function. Since summation of concave functions is also concave, the objective function f is also concave along any edge on $\mathbf{G}$.

P-Median Problem

In P-Median Problem, vertices are assigned to clusters with the closest cluster center minimizing (2.9).

In 1965, Hakimi generalized his previous proof to P-Median Problem, and he proved that $\mathbf{V}$ contains the optimal solution in P-Median Problem [23]. In his proof, he separates vertices according to clusters they are assigned to. Then, he separates the problem to p 1-Median Problems. As proof in [22] implies, in each separate problem, the optimal location is on vertices. When solution of each 1-Median Problem is found, solutions are combined to form a solution to the P-Median Problem. Since assignments of vertices to clusters may change, assignments are arranged again. With this solution where centers are located on the vertices, the objective function value is less than the solution where centers are located on edges.

Another approach to this problem could be made by analyzing behavior of the objective function along the edges. Suppose we have a network with *p* cluster centers located. Let x_c and x_k be the closest and second closest cluster centers to vertex v_i, respectively. An illustration is given in Figure 4.4. If we keep locations of *p-1* cluster centers fixed and move one cluster center (let us say x_c) along the edge (v_p, v_q), vertex v_i that was assigned to cluster 1 may be assigned to cluster with the center x_k. If the assignment of v_i changes, its objective function component also changes its behavior. In the case when v_i is assigned to another cluster center x_k, since location of x_k is fixed, the objective function remains constant. Therefore, the objective function component of v_i could be observed as in Figure 4.5. In the figure, (a) is the case that shortest path to x_c passes from v_p in the interval $[v_p, a_i]$, and v_i has been assigned to cluster with center x_k in $[a_i, v_q]$. (a) is the case that v_i is assigned to x_k in $[v_p, a_i]$, and as x_c is moved towards v_q, v_i has been assigned to cluster x_c. In (c), arc bottleneck point b_i has been observed. As x_c is moved from v_p towards v_q, before arriving b_i, v_i is assigned to cluster with center x_k at the assignment bottleneck point a_i^p. Therefore, the objective function becomes a constant value. After passing the bottleneck point b_i, the shortest path to x_c starts to pass from v_q, $d(v_i, x_1)$ starts to decrease and v_i is assigned back to x_c at the assignment bottleneck point a_i^q. As a result, when there

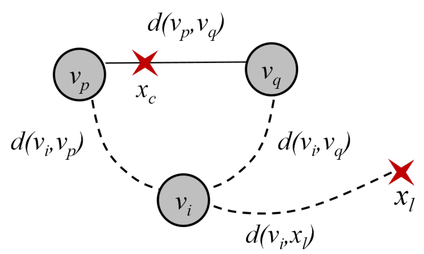

Figure 4.4: A visualization of a part of graph $\mathbf{G}$ with 2 closest cluster centers

are multiple centers, additional to arc bottleneck point that has been observed in 1-median case, assignment bottleneck point is observed. Even when this point has been observed, piecewise concavity is valid for f_i. Since summation of concave functions are concave, the objective function f is concave, and as Levy found [24], x_c will be located on vertices.

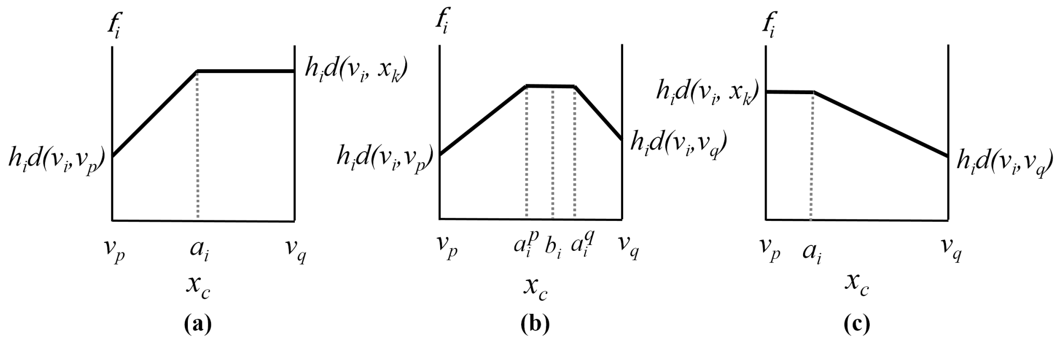

Figure 4.5: Objective function component for v_i (denoted as f_i) when x_c is moved along the edge (v_p, v_q) and x_k is the second closest cluster center to v_i

4.1.2 Sum of Squares Clustering (SSC) Problem on Networks

SSC problem differs from P-Median Problem in that it uses sum of squared distances in objective function instead of sum of distances. SSC problem is defined as

$$\textit{minimize } f(\mathbf{X}) = \sum_{i=1}^{n} h_i \min_{k=1,\dots,p} \{d(v_i, x_k)^2\}$$

$$\text{subject to}$$

$$x_k \in G \quad \forall\, k = 1, ..., p,$$

where x_k is the decision variable for center location of cluster center k, and h_i is a nonnegative constant for weight of v_i. As in P-Median Problem, because of the hard assignment, each vertex will be assigned to one of the centers. For SSC problem, Carrizosa et. al. observed that the optimal solution could be observed on not only $\mathbf{V}$, but also $\mathbf{E}$. Therefore, different from P-Median Problem, location of cluster centers are restricted to $\mathbf{G}$. In this subsection, this property is analyzed in more detail.

SSC on Networks with a Single Cluster

In SSC problem, same as P-Median Problem, when there is one cluster, each vertex will be assigned to that cluster. As in the example shown in Figure 4.1, we consider a line graph with 4 vertices. Now in SSC case, the objective function for this example is

$$f = (a+y)^2 + y^2 + (l-y)^2 + (b+l-y)^2.$$

This function is continuous and twice differentiable. First and second order derivatives with respect to y are

$$\frac{df}{dy} = 8y + 2a - 2l - 2b - 2l, \tag{4.3}$$

$$\frac{d^2f}{dy^2} = 8. \tag{4.4}$$

(4.4) is positive, which shows that f is a convex function of y. In order to find the minimum value, we need to set value of (4.3) to 0 and solve it for y. Then,

$$y = \frac{b+2l-a}{4},$$

which minimizes the objective function. Regarding the value of y, the following cases could be seen.

- If $y \in (0, l)$, the optimal center location is on the edge (v_2, v_3) but not on vertices.
- If $y = 0$, the optimal center location is vertex v_2.
- If $y = l$, the optimal center location is vertex v_3.
- If $y \in (0, -a)$, the optimal center location is on the edge (v_1, v_2) but not on vertices.
- If $y = -a$, the optimal center location is vertex v_1.
- If $y \in (l, b+l)$, the optimal center location is on the edge (v_3, v_4) but not on vertices.
- If $y = b + l$, the optimal center location is vertex v_4.

Assuming that the center will be located on the edge (v_2, v_3), these cases could be interpreted as the following: if $y \leq 0$, the best location is vertex v_2. If $y \geq l$, the best location is vertex v_3. In the other cases, the optimal center location could be a point not on the vertices, but on the edge (v_2, v_3).

In a more generalized version of the line graph in Figure 4.2 that contains n vertices in which cluster center x_c is to be located. Let x_c be located on the edge (v_p, v_q). The objective function will be

$$f = \sum_{i=1}^{n} h_i d(v_i, x_c)^2. \tag{4.5}$$

We know that

$$d(v_i, x_c) = \min \left\{ d(v_i, v_p) + d(v_p, x_c), d(v_i, v_q) + d(v_q, x_c) \right\}.$$

Assume that set of vertices $i = 1, ..., n$ have been arranged such that

$$d(v_{i_j}, x_c) = d(v_{i_j}, v_p) + d(v_p, x_c), for\ j = 1, ..., r,$$
$$d(v_{i_j}, x_c) = d(v_{i_j}, v_q) + d(v_q, x_c), for\ j = r+1, ..., n.$$

Then, the objective function could be written as

$$f = \textstyle\sum_{j=1}^{r} h_{i_j}(d(v_{i_j}, v_p) + d(v_p, x_c))^2 + \sum_{j=r+1}^{n} h_{i_j}(d(v_{i_j}, v_q) + d(v_q, x_c)). \tag{4.6}$$

Substitute $d(v_q, x_c) = d(v_p, v_q) - d(v_p, x_c)$ the objective function (4.6) is

$$f = \textstyle\sum_{j=1}^{r} h_{i_j}(d(v_{i_j}, v_p) + d(v_p, x_c))^2 + \sum_{j=r+1}^{n} h_{i_j}(d(v_{i_j}, v_q) + d(v_p, v_q) - d(v_p, x_c))^2.$$

This function is continuous and twice differentiable. First and second order derivatives are

$$\begin{aligned} \frac{\partial f}{\partial x_c} =& \sum_{j=1}^{n} h_{i_j} 2d(v_p, x_c) \\ &+ \sum_{j=1}^{r} 2h_{i_j} d(v_{i_j}, v_p) - \sum_{j=r+1}^{n} 2h_{i_j}(d(v_{i_j}, v_q) + d(v_p, v_q)) && (4.7) \\ \frac{\partial^2 f}{\partial x_c^2} =& \sum_{j=1}^{n} 2h_{i_j}. && (4.8) \end{aligned}$$

(4.8) is positive, which shows that z is a convex function of $d(v_p, x_c)$. To find the minimum value, we are zeroing (4.7) and solve it for $d(v_p, x_c)$. Then,

$$d(v_p, x_c) = \frac{\sum_{i=r+1}^{n} h_{i_j}(d(v_{i_j}, v_q) + d(v_p, v_q)) - \sum_{i=1}^{r} h_{i_j} d(v_{i_j}, v_p)}{\sum_{i=1}^{n} h_{i_j}}.$$

From this equation, given that the center is to be located on (v_p, v_q), the following interpretations could be made. If $d(v_p, x_c) \leq 0$, x_c will be located on vertex v_p. If

$d(v_p, x_c) \geq d(v_p, v_q)$, x_c will be located on vertex v_q. For other values of $d(v_p, x_c)$, x_c will be located on the edge (v_p, v_q). Hence, to find the optimal solution, one should not restrict solution space to vertices since optimal solution could be on edges in SSC problem on a line graph.

Now we can generalize our results for a network **G** with one cluster. Let us assume that the cluster center is on edge (v_p, v_q) and f_i denotes the objective function component of vertex v_i. There are three cases of f_i as shown in Figure 4.6. In (a) and (c), the shortest path to the center passes from v_p and v_q, respectively. In (b), when $x_c \in [v_p, b_i]$ where b_i is the arc bottleneck point, the shortest path passes from v_p; otherwise, the shortest path passes from v_q. The function is piecewise and both the function in $(0, b_i)$ and the function in (b_i, l) are convex by second derivative test. Each piece of f_i is convex and increasing with the distance. Because of the convexity of each f_i, f is also convex, ehich implies that f may contain a local minimum along the edge. Therefore, the optimal solution could be found on the edges.

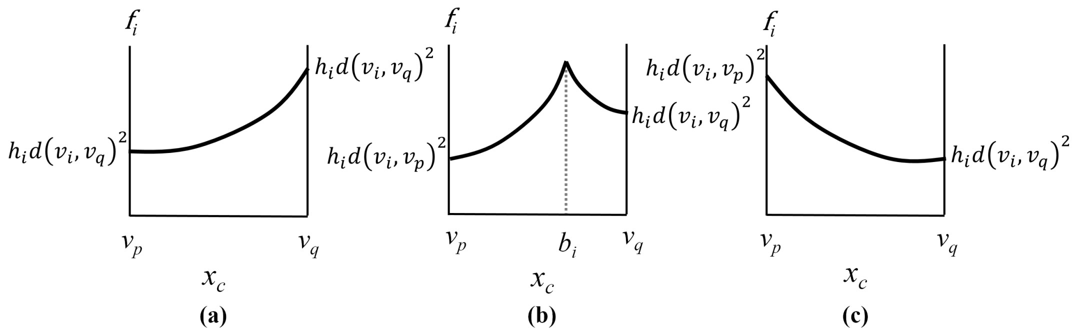

Figure 4.6: Objective function component for v_i (denoted as f_i) in SSC problem with single cluster when x_c is moved along the edge (v_p, v_q)

Theorem 4.1.1. *In the SSC Problem with single center, the optimal center could be located to interior point of an edge.*

Proof. In order to prove this theorem, we will take objective function value of a vertex which has the best objective function value among vertex set V. Then, we will show that under certain conditions, an interior point along the edge will have a lower objective value. Let objective function value at vertex v_p be f^p. Then,

$$f^p = \sum_{i=1}^{n} h_i d(v_i, v_p)^2.$$

Assume that the set of points have been arranged such that for points $j = 1, ..., r$ the shortest path to v_p does not contain the edge (v_p, v_q), and for points $j = r+1, ..., n$, the shortest path to v_p contains the edge (v_p, v_q), that is, $d(v_{i_j}, v_p) = d(v_{i_j}, v_q) + d(v_p, v_q)$. Then,

$$f_p = \sum_{i=1}^{r} h_{i_j} d(v_{i_j}, v_p)^2 + \sum_{i=r+1}^{n} h_{i_j} (d(v_{i_j}, v_q) + d(v_p, v_q))^2.$$

There exists a point x on the edge (v_p, v_q) at which the same partitioning is valid. Then, the objective function value at x is

$$f^x = \sum_{i=1}^{r} h_{i_j} (d(v_{i_j}, v_p) + d(v_p, x))^2 + \sum_{i=r+1}^{n} h_{i_j} (d(v_{i_j}, v_q) + d(v_p, v_q) - d(v_p, x))^2.$$

Rearranging this expression, we have

$$f^x = f^p + d(v_p, x)^2 \left[\sum_{i=1}^{n} h_{i_j}\right] + 2d(v_p, x) \left[\sum_{i=1}^{r} h_{i_j} d(v_{i_j}, v_p) - \sum_{i=r+1}^{n} h_{i_j} (d(v_{i_j}, v_q) + d(v_p, v_q))\right].$$

Let

$$a = \sum_{i=1}^{n} h_{i_j}, b = \sum_{i=1}^{r} h_{i_j} d(v_{i_j}, v_p) - \sum_{i=r+1}^{n} h_{i_j} (d(v_{i_j}, v_q) + d(v_p, v_q)).$$

Then, we have

$$f^x = f^p + ad(v_p, x)^2 + 2bd(v_p, x). \tag{4.9}$$

On the right-hand side, the expression after f^p is a quadratic function of $d(v_p, x)$, that is, $f(x) = a^2 + 2bx$. From (4.9), we can say that if $f(x) \leq 0$, $z^x <= f^p$. If $f(x) > 0$, $f^x > f^p$. This is possible when $d(v_p, x) \in [0, -2b/a]$. In order for x to have a nonempty interval, $-2b/a \geq 0$. Since a is nonnegative, this is possible if $b \leq 0$. Hence, the following condition must hold.

$$\sum_{i=1}^{r} h_{i_j} d(v_{i_j}, v_p) \leq \sum_{i=r+1}^{n} h_{i_j} (d(v_{i_j}, v_q) + d(v_p, v_q))$$

□

SSC on Networks with p Clusters

In SSC problem with p clusters, each vertex is assigned to the cluster whose center has the minimum distance, and cluster centers are located such that the sum of squared distances multiplied by a nonnegative weight is minimized.

If objective function is analyzed, it could be seen that the objective function is summation of second degree polynomials. Suppose we have *p* centers on $\mathbf{G}$. If we fix *p-1* centers' locations, take a cluster center x_c and move this center along the edge (v_p, v_q), as observed in P-Median Problem, assignments of the vertices to clusters may change, i.e., vertices could be assigned to their second closest clusters. Therefore, in an example given in Figure 4.4 and described above, f_i, contribution of v_i to objective function f, could behave as in Figure 4.7. As explained in Subsection 4.1.1 for P-Median Problem, f_i is piecewise at the arc bottleneck points and assignment bottleneck points. If assignment does not change, behavior of f_i changes at arc bottleneck point. If assignment changes, behavior of f_i changes at assignment bottleneck point(s).

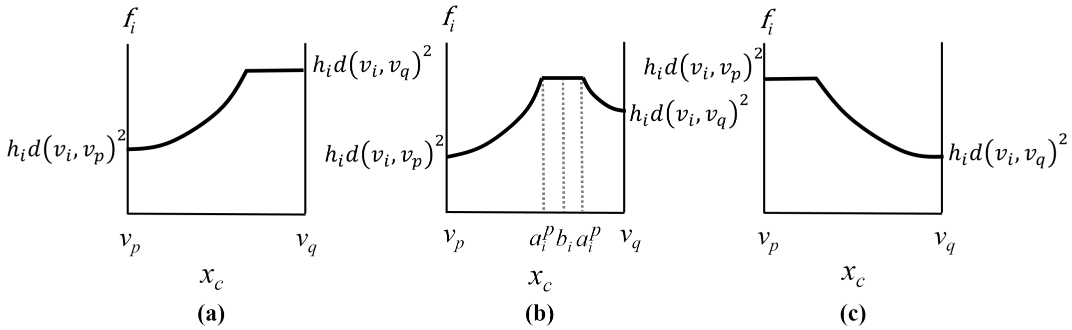

Figure 4.7: Objective function component for v_i (denoted as f_i) in SSC problem with p clusters when x_c is moved along the edge (v_p, v_q) and x_k is the second closest cluster center to v_i

Theorem 4.1.2. *Let* $\mathbf{V}^*$ *be a set of p vertices* $\{v_1^*, v_2^*, ..., v_p^*\}$ *which is the optimal solution among all possible* $\mathbf{V}$ *sets. There exists a subset* $\mathbf{X}^* \in \mathbf{G}$ *containing s (s* $\leq$ *p) centers located on edges and the remaining centers at vertices* $\{x_c, x_2, ..., x_s, \}$

$\{v_{s+1}^*, v_{s+2}^*, ..., v_p^*\}$, *such that*

$$\sum_{k=1}^{n} h_i d(v_i, \mathbf{V}^*)^2 \geq \sum_{k=1}^{n} h_i d(v_i, \mathbf{X})^2$$

Proof. Let $\{v_1^*, v_2^*, ..., v_p^*\}$ be set of points in $\mathbf{V}^*$. If these points are rearranged such that

$$\begin{aligned}
d(v_{i_j}, \mathbf{V}^*) =& d(v_{i_j}, v_1^*) \quad \forall j = 1, ...n_1, \\
d(v_{i_j}, \mathbf{V}^*) =& d(v_{i_j}, v_2^*) \quad \forall j = n_1 + 1, ...n_2, \\
& ... \\
d(v_{i_j}, \mathbf{V}^*) =& d(v_{i_j}, v_p^*) \quad \forall j = n_{p-1} + 1, ...n_p = n,
\end{aligned}$$

the objective function could be written as

$$f \;=\; \sum_{j=1}^{n_1} h_{i_j} d(v_{i_j}, v_1^*)^2 \;+\; \sum_{j=n_1+1}^{n_2} h_{i_j} d(v_{i_j}, v_2^*)^2 \;+\; ... \;+\; \sum_{j=n_{p-1}+1}^{n_p} h_{i_j} d(v_{i_j}, v_p^*)^2$$

Let

$$\begin{aligned}
f_1 =& \sum_{j=1}^{n_1} h_{i_j} d(v_{i_j}, v_1^*)^2 \\
f_2 =& \sum_{j=n_1+1}^{n_2} h_{i_j} d(v_{i_j}, v_2^*)^2 \\
& ... \\
f_p =& \sum_{j=n_{p-1}+1}^{n_p} h_{i_j} d(v_{i_j}, v_p^*)^2.
\end{aligned}$$

Define $h'_{i_j} = h'_{i_j}$ for $j = 1, ..., n_1$ and $h'_{i_j} = 0$ for $j = n_1 + 1, ..., n_p$, we have

$$f_1 = \sum_{j=1}^{n} h_{i_j} d(v_{i_j}, v_1^*)^2.$$

In previous theorem, we have shown that given a condition, there exists an interior point x_c on an edge adjacent to v_1^* such that

$$f_1 \geq \sum_{j=1}^{n} h'_{i_j} d(v_{i_j}, x_c)^2,$$

which could be written as

$$f_1 \geq \sum_{j=1}^{n_1} h_{i_j} d(v_{i_j}, x_c)^2.$$

Assume that for cluster centers 1,...,s, this condition is satisfied. Then,

$$\begin{aligned} f_1 &\geq \sum_{j=1}^{n_1} h_{i_j} d(v_{i_j}, x_c)^2, \\ f_2 &\geq \sum_{j=n_1+1}^{n_2} h_{i_j} d(v_{i_j}, x_2)^2, \\ &\dots \\ f_s &\geq \sum_{j=n_{s-1}+1}^{n_s} h_{i_j} d(v_{i_j}, x_s)^2. \end{aligned}$$

Adding both sides of the inequalities, we have

$$\begin{aligned} f \geq \sum_{j=1}^{n_1} h_{i_j} d(v_{i_j}, x_c)^2 + \sum_{j=n_1+1}^{n_2} h_{i_j} d(v_{i_j}, x_2)^2 + \dots + \sum_{j=n_{s-1}+1}^{n_s} h_{i_j} d(v_{i_j}, x_s)^2 \\ + \sum_{j=n_s+1}^{n_{s+1}} h_{i_j} d(v_{i_j}, v^*_{s+1})^2 + \sum_{j=n_{s+1}+1}^{n_{s+2}} h_{i_j} d(v_{i_j}, v^*_{s+2})^2 \\ + \dots + \sum_{j=n_{p-1}+1}^{n_p} h_{i_j} d(v_{i_j}, v^*_p)^2 \end{aligned} \tag{4.10}$$

Let the new set of centers $X = \{x_c, x_2, ..., x_s, v^*_{s+1}, v^*_{s+2}, ..., v^*_p\}$. After changing locations of cluster centers, assignments of vertices to centers may change. Therefore, we have

$$\begin{aligned} \sum_{j=1}^{n_1} h_{i_j} d(v_{i_j}, x_c)^2 + \sum_{j=n_1+1}^{n_2} h_{i_j} d(v_{i_j}, x_2)^2 + \dots + \sum_{j=n_{s-1}+1}^{n_s} h_{i_j} d(v_{i_j}, x_s)^2 \\ + \sum_{j=n_s+1}^{n_{s+1}} h_{i_j} d(v_{i_j}, v^*_{s+1})^2 + \sum_{j=n_{s+1}+1}^{n_{s+2}} h_{i_j} d(v_{i_j}, v^*_{s+2})^2 \\ + \dots + \sum_{j=n_{p-1}+1}^{n_p} h_{i_j} d(v_{i_j}, v^*_p)^2 \geq \sum_{j=1}^{n} h_i d(v_i, \mathbf{X})^2. \end{aligned} \tag{4.11}$$

Combining (4.10) and (4.11), we have

$$\sum_{i \in \mathbf{V}} h_i d(v_i, \mathbf{V}^*)^2 \geq \sum_{i \in \mathbf{V}} h_i d(v_i, \mathbf{X})^2.$$

$\square$

As a result, contrary to P-Median Problem, in SSC problem, restricting the cluster center locations as vertices could prevent one from finding the optimal solution on the graph.

4.2 Clustering Problems on Networks with Soft Assignment

In this section, two clustering problems that perform soft assignment will be discussed. In soft assignment, each vertex is assigned to all clusters with a probability. There are two soft clustering problems defined in the scope of this study. Both of these problems have been studied on the plane in the literature, and to the best of our knowledge, they have not been studied on networks before. These two problems are called as Probabilistic Distance Clustering (PD-Clustering) and Fuzzy Clustering (FC). These problems differ in the objective functions and membership functions they use. As a result of this difference, they have different properties, which will be discussed in further detail.

4.2.1 Probabilistic Distance Clustering (PD-Clustering) Problem on Networks

On network, PD-Clustering Problem is defined as

$$\textit{minimize} \quad f(\mathbf{X}) = \sum_{i=1}^{n} \sum_{k=1}^{p} p_{ik}^2 d(v_i, x_k) \tag{4.12}$$

$$\begin{aligned} &\text{subject to} \\ &\sum_{k=1}^{p} p_{ik} = 1 \quad \forall\, i = 1, ..., n, \\ &p_{ik} \geq 0 \quad \forall\, i = 1, ..., n,\ k = 1, ..., p, \\ &x_k \in V \quad \forall\, k = 1, ..., p, \end{aligned} \tag{4.13}$$

where x_k is the location of cluster center k and p_{ik} is the membership value of v_i to cluster k. For each vertex, summation of memberships to all clusters must be equal to 1. As shown in [29] by Iyigun and Ben-Israel, by using Lagrangian Method, keeping all x_k fixed, membership function is

$$p^*_{ik} = \frac{1}{\sum_{l=1}^{p} \frac{d(v_i,x_k)}{d(v_i,x_l)}}. \tag{4.14}$$

When this problem is analyzed, it is observed that the optimal center locations are on $\mathbf{V}$, which will be proven in this subsection.

PD-Clustering on Networks with a Single Cluster

In PD-Clustering Problem, when there is one cluster, each vertex will be have a membership value of 1 to that cluster. As a result, the problem becomes similar to 1-median problem. f_i, the objective function component of v_i, is as illustrated in Figure 4.3 in the example illustrated in Figure 4.2. It is linear and piecewise concave, and its behavior changes at arc bottleneck point if x_c is moved along an edge (v_p, v_q). If there is an arc bottleneck point on an edge as in (b), f_i is linear and piecewise concave along the edge such that it has its maximum at the arc bottleneck point. If there are no arc bottleneck points as in (a) and (c), the distance function is linear. Summation of piecewise concave and linear functions is linear and piecewise concave as well, which is the objective function (4.12). Therefore, locating the center to an interior point of an edge will lead higher objective function values. The theorem and its proof is given below.

Theorem 4.2.1. *In single center PD-Clustering Problem,* $\mathbf{V}$ *contains the set of optimal solutions.*

Proof. To prove this theorem, it will be shown that an interior point x on an edge (v_p, v_q) could not have a lower objective value than the vertex v_p which has the best objective value among vertex set $\mathbf{V}$. Let objective function value at vertex v_p be f^p.

Then,

$$f^p = \sum_{i=1}^{N} p_i^2 d(v_i, v_p).$$

Assume that the set of vertices arranged as the ones whose shortest path to v_p contains the edge (v_p, v_q) or not. Let $v_{i_j}, j = 1, ..., r$ show the vertices that does not contain the edge (v_p, v_q), and $j = r+1, ..., n$ show the ones that contains the edge (v_p, v_q), that is, $d(v_{i_j}, v_p) = d(v_{i_j}, v_q) + d(v_p, v_q)$. Then,

$$f^p = \sum_{j=1}^{r} p_{i_j}^2 d(v_{i_j}, v_p) + \sum_{j=r+1}^{N} p_{i_j}^2 (d(v_{i_j}, v_q) + d(v_p, v_q)).$$

There exists a center x on the edge (v_p, v_q) at which the same arrangement is valid. Then, the objective function value at x is

$$f^x = \sum_{j=1}^{r} p_{i_j}^2 (d(v_{i_j}, v_p) + d(v_p, x)) + \sum_{j=r+1}^{N} p_{i_j}^2 ((d(v_{i_j}, v_q) + d(v_p, v_q) - d(v_p, x))).$$

Rearranging this expression, we have

$$f^x = f^p + d(v_p, x) \left[\sum_{j=1}^{r} p_{i_j}^2 - \sum_{j=r+1}^{n} p_{i_j}^2 \right].$$

$$\sum_{j=1}^{r} p_{i_j}^2 \geq \sum_{j=r+1}^{n} p_{i_j}^2 \implies f^p \leq f^x. \tag{4.15}$$

Suppose that

$$\sum_{j=1}^{r} p_{i_j}^2 < \sum_{j=r+1}^{n} p_{i_j}^2. \tag{4.16}$$

Since $d(v_p, v_q)$ is a positive constant, multiplying both sides with $d(v_p, v_q)$, we have

$$d(v_p, v_q) \sum_{j=1}^{r} p_{i_j}^2 < d(v_p, v_q) \sum_{j=r+1}^{n} p_{i_j}^2.$$

We may rewrite f^p as

$$f^p = \sum_{j=1}^{r} p_{i_j}^2 d(v_{i_j}, v_p) + \sum_{j=r+1}^{N} p_{i_j}^2 d(v_{i_j}, v_q) + \sum_{j=r+1}^{N} p_{i_j}^2 d(v_p, v_q).$$

By (4.16), we may write

$$f^p > \sum_{j=1}^{r} p_{i_j}^2 d(v_{i_j}, v_p) + \sum_{j=r+1}^{N} p_{i_j}^2 d(v_{i_j}, v_q) + \sum_{j=1}^{r} p_{i_j}^2 d(v_p, v_q).$$

Right-hand side of this inequality is an upper bound to the objective function value at v_q. Hence,

$$f^p > f^q,$$

which contradicts with v_p's having the minimum objective value among all vertices. This implies that f^x will always be greater than f^p. Therefore, the optimal location will always be on a vertex. $\square$

As stated previously, Hakimi generalized this case for P-Median Problem and proved that the optimal center locations are always on vertices in [23]. However, P-Median Problem is not similar to PD-Clustering since P-Median assigns each vertex to the closest cluster while the PD-Clustering calculates membership value for each vertex-cluster pair. In the following subsection, PD-Clustering Problem with p clusters will be analyzed.

PD-Clustering on Networks with p Clusters

As stated previously, when there are p clusters, PD-Clustering works with membership values which depend on location of centers. In this subsection, it will be proven that in the optimal solution, centers of a PD-Clustering Problem with p clusters on a network will be on vertices.

For the sake of simplicity, suppose we have two clusters. If (4.14) is evaluated for this case, membership value of vertex i will be

$$p_{i1} = \frac{d(v_i, x_2)}{d(v_i, x_c) + d(v_i, x_2)}, \qquad p_{i2} = \frac{d(v_i, x_c)}{d(v_i, x_c) + d(v_i, x_2)}. \tag{4.17}$$

For a graph $\mathbf{G}$ with two clusters as in Figure 4.2, keeping x_2 fixed and moving x_c on the edge (v_p, v_q), change in the membership functions p_{i1} and p_{i2} has been visualized in Figure 4.8. Although x_2 has a fixed location, it is affected from the location change of x_c. In (a), (b) and (c), as x_c moves towards arc bottleneck point b_i, p_{i1} decreases since distance increases and reaches the maximum at b_i. As p_{i1} decreases, p_{i2} increases. In (a), even at the arc bottleneck point, $d(v_i, x_1)$ is less than $d(v_i, x_2)$;

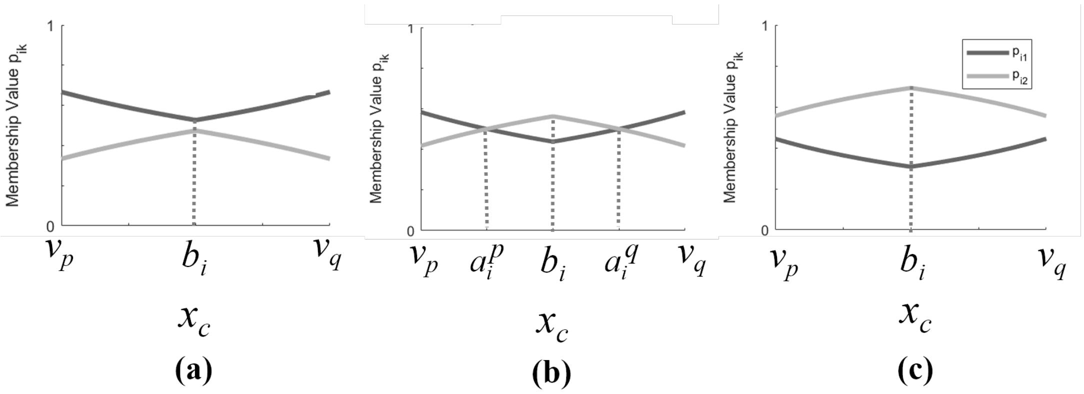

Figure 4.8: Membership function p_{ik} of PD-Clustering with 2 clusters

therefore, p_{i2} increases but it cannot be greater than p_{i1}. In (c), the contrast occurs. $d(v_i, x_2)$ is less than $d(v_i, x_1)$ even when x_c is located on endpoints v_p or v_q. Therefore, p_{i1} is always less than p_{i2}. In (b), when $x_c \in [v_p, a_i^p]$, p_{i1} is greater than p_{i2}. When $x_c \in [a_i^p, a_i^q]$, p_{i2} is greater than p_{i1} since $d(v_i, x_2)$ is less than $d(v_i, x_1)$. Lastly, as $x_c \in [a_i^q, v_q]$, again, $d(v_i, x_1)$ decreases and becomes less than $d(v_i, x_2)$. Therefore, p_{i1} is greater than p_{i2}. In (b), assignment bottleneck points could be observed as the points where p_{i1} =p_{i2}.

If (4.17) is substituted in (4.12), the objective function will be

$$f(\mathbf{X}) = \sum_{i=1}^{n} \frac{d(v_i, x_c)d(v_i, x_2)}{d(v_i, x_c) + d(v_i, x_2)}.$$

For three clusters, the resulting membership values will be

$$p_{i1} = \frac{d(v_i, x_2)d(v_i, x_3)}{d(v_i, x_c)d(v_i, x_2) + d(v_i, x_2)d(v_i, x_3) + d(v_i, x_c)d(v_i, x_3)},$$
$$p_{i2} = \frac{d(v_i, x_c)d(v_i, x_3)}{d(v_i, x_c)d(v_i, x_2) + d(v_i, x_2)d(v_i, x_3) + d(v_i, x_c)d(v_i, x_3)},$$
$$p_{i3} = \frac{d(v_i, x_c)d(v_i, x_2)}{d(v_i, x_c)d(v_i, x_2) + d(v_i, x_2)d(v_i, x_3) + d(v_i, x_c)d(v_i, x_3)}.$$

With the same manner, the objective function could be written as

$$f(\mathbf{X}) = \sum_{i=1}^{n} \frac{d(v_i, x_c)d(v_i, x_2)d(v_i, x_3)}{d(v_i, x_c)d(v_i, x_2) + d(v_i, x_2)d(v_i, x_3) + d(v_i, x_c)d(v_i, x_3)}.$$

For the problem with p clusters, the objective function is

$$f = \sum_{i=1}^{n} \frac{\prod_{k \in K} d(v_i, x_k)}{\sum_{k \in K} \prod_{l \neq k} d(v_i, x_l)}. \tag{4.18}$$

Since we assume that location of x_k is fixed for $k = 2, ..., P$, we could separate constant components of each vertex as K_i and write the objective function (4.18) as in (4.19).

$$K_i = \frac{\prod_{k=2}^{p} d(v_i, x_k)}{\sum_{k=2}^{p} \prod_{l \neq k} d(v_i, x_l)} \rightarrow \qquad f(x_c) = \sum_{i=1}^{n} \frac{d(v_i, x_c) K_i}{d(v_i, x_c) + K_i} \tag{4.19}$$

Let the memberships p_{i1} of each point considering the location of cluster center 1 be

$$p_{i1} = \frac{K_i}{d(v_i, x_c) + K_i}. \tag{4.20}$$

Then, objective function (4.19) could be simplified as

$$f(x_c) = \sum_{i=1}^{n} d(v_i, x_c) p_{i1}.$$

f_i, contribution of vertex v_i to the function (4.19), is continuous and twice differentiable. First and second order derivatives are

$$\begin{aligned} \frac{df_i}{dx_c} &= \frac{K_i^2}{(K_i + d(v_i, x_c))^2} \\ \frac{d^2 f_i}{dx_c^2} &= \frac{-2K_i^2}{(K_i + d(v_i, x_c))^3} \end{aligned} \tag{4.21}$$

With the second derivative test, since (4.21) is always negative, we can conclude that f_i is concave. Let $\mathbf{G}$ be a graph with p clusters. Keeping *p-1* clusters fixed and moving one cluster center (let us say x_c) along the edge $(v_p, v - q)$, f_i function could be observed as given in Figure 4.9. In (a) and (c), the shortest path from v_i to x_c passes from v_p and v_q, respectively. In (b), when $x_c \in [v_p, b_i]$, f_i increases. When $x_c \in [b_i, v_q]$, f_i decreases with the decreasing distance. In (b), f_i is piecewise concave. Different from hard assignment problems, piecewiseness occurs at only arc bottleneck points. Since each f_i is concave or piecewise concave, f, summation of $f_i \, \forall \, i = 1, ..., n$, is also concave.

Theorem 4.2.2. *In Probabilistic Distance Clustering Problem on Networks, a cluster center is always at a vertex of a network* $\mathbf{G} = (\mathbf{E}, \mathbf{V})$.

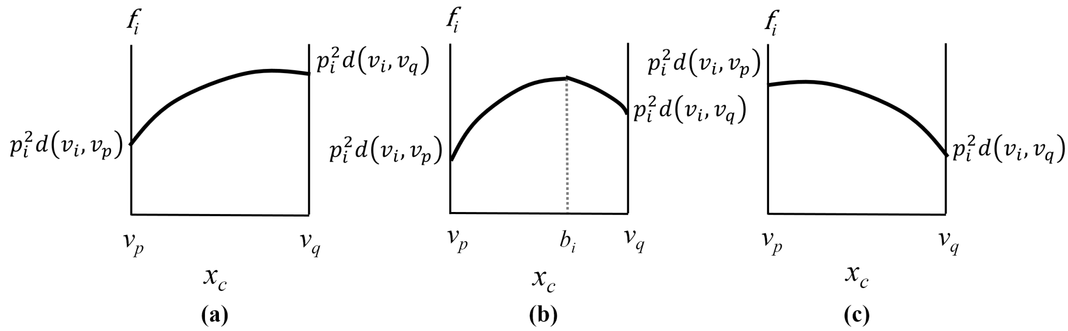

Figure 4.9: Objective function component for v_i (denoted as f_i) in PD-Clustering Problem with p clusters when x_c is moved along the edge (v_p, v_q) and x_k is the second closest cluster center to v_i

Proof. To prove this theorem, it will be shown that keeping $x_k \; \forall\, k = 1, ..., p - \{c\}$ fixed, x_c will always be located on a vertex. Let y_0 be an arbitrary point on the edge $(v_p, v_q) \in \mathbf{E}$ and $y_0 \notin V$. There exists a vertex v_m such that

$$\sum_{i=1}^{n} \frac{d(v_i, y_0)K_i}{d(v_i, y_0) + K_i} \geq \sum_{i=1}^{n} \frac{d(v_i, v_m)K_i}{d(v_i, v_m) + K_i}.$$

We know that

$$d(v_i, y_0) = \min \{d(v_i, v_p) + d(v_p, y_0), d(v_i, v_q) + d(v_q, y_0)\} .$$

Assume that the set of points have been arranged such that

$$d(v_{i_j}, y_0) = d(v_{i_j}, v_p) + d(v_p, y_0), for\ j = 1, ..., r,$$
$$d(v_{i_j}, y_0) = d(v_{i_j}, v_q) + d(v_q, y_0), for\ j = r + 1, ..., N.$$

Then, the objective function could be written as

$$\sum_{j=1}^{n} \frac{d(v_{i_j}, y_0)K_{i_j}}{d(v_{i_j}, y_0) + K_{i_j}} = \sum_{j=1}^{r} \frac{(d(v_{i_j}, v_p) + d(v_p, y_0))K_{i_j}}{d(v_{i_j}, v_p) + d(v_p, y_0) + K_{i_j}} + \sum_{j=r+1}^{n} \frac{(d(v_{i_j}, v_q) + d(v_q, y_0))K_{i_j}}{d(v_{i_j}, v_q) + d(v_q, y_0) + K_{i_j}}. \quad (4.22)$$

Since we have two vertices v_p and v_q connected by the edge which contains y_0, either v_p or v_q is a better solution. We will consider two cases each implying that one of the vertices is a better solution.

Substitute $d(v_q, y_0) = d(v_p, v_q) - d(v_p, y_0)$ the objective function in (4.22) is

$$f = \sum_{j=1}^{n} \frac{d(v_{i_j}, y_0)K_{i_j}}{d(v_{i_j}, y_0) + K_{i_j}} = \sum_{j=1}^{r} \frac{(d(v_{i_j}, v_p) + d(v_p, y_0))K_{i_j}}{d(v_{i_j}, v_p) + d(v_p, y_0) + K_{i_j}} + \sum_{j=r+1}^{n} \frac{(d(v_{i_j}, v_q) + d(v_p, v_q) - d(v_p, y_0))K_{i_j}}{d(v_{i_j}, v_q) + d(v_p, v_q) - d(v_p, y_0) + K_{i_j}}.$$

Let

$$f = f_1 + f_2,$$

where

$$f_1 = \sum_{j=1}^{r} \frac{(d(v_{i_j}, v_p) + d(v_p, y_0))K_{i_j}}{d(v_{i_j}, v_p) + d(v_p, y_0) + K_{i_j}}, \tag{4.23}$$

$$f_2 = \sum_{j=r+1}^{n} \frac{(d(v_{i_j}, v_q) + d(v_p, v_q) - d(v_p, y_0))K_{i_j}}{d(v_{i_j}, v_q) + d(v_p, v_q) - d(v_p, y_0) + K_{i_j}}. \tag{4.24}$$

Multiply both the numerator and denominator of each term of the summation of f_1 in (4.23) by $d(v_{i_j}, v_p) + K_{i_j}$, then

$$f_1 = \sum_{j=1}^{r} \left[\frac{d(v_{i_j}, v_p)K_{i_j}}{d(v_{i_j}, v_p) + K_{i_j}} + \frac{d(v_p, y_0)K_{i_j}^2}{(d(v_{i_j}, v_p) + d(v_p, y_0) + K_{i_j})(d(v_{i_j}, v_p) + K_{i_j})} \right].$$

Similarly, multiply both the numerator and denominator of each term of the summation of f_2 in (4.24) by $(d(v_{i_j}, v_q) + d(v_p, v_q) + K_{i_j})$, then

$$f_2 = \sum_{j=r+1}^{n} \left[\frac{(d(v_{i_j}, v_q) + d(v_p, v_q))K_{i_j}}{d(v_{i_j}, v_q) + d(v_p, v_q) + K_{i_j}} - \frac{d(v_p, y_0)K_{i_j}^2}{(d(v_{i_j}, v_q) + d(v_p, v_q) - d(v_p, y_0) + K_{i_j})(d(v_{i_j}, v_q) + d(v_p, v_q) + K_{i_j})} \right]. \tag{4.25}$$

By triangle inequality, we have $d(v_{i_j}, v_q) + d(v_p, v_q) \geq d(v_{i_j}, v_p)$. Then

$$\sum_{j=r+1}^{n} \frac{(d(v_{i_j}, v_q) + d(v_p, v_q))K_{i_j}}{d(v_{i_j}, v_q) + d(v_p, v_q) + K_{i_j}} \geq \sum_{j=r+1}^{n} \frac{d(v_{i_j}, v_p)K_{i_j}}{d(v_{i_j}, v_p) + K_{i_j}}. \tag{4.26}$$

Substitute right-hand side of (4.26) in (4.25), we have

$$f_2 \geq f_2' = \sum_{j=r+1}^{n} \left[\frac{d(v_{i_j}, v_p)K_{i_j}}{d(v_{i_j}, v_p) + K_{i_j}} - \frac{d(v_p, y_0)K_{i_j}^2}{(d(v_{i_j}, v_q) + d(v_p, v_q) - d(v_p, y_0) + K_{i_j})(d(v_{i_j}, v_q) + d(v_p, v_q) + K_{i_j})} \right]. \tag{4.27}$$

Since $d(v_{i_j}, v_q) + d(v_p, v_q) \geq d(v_{i_j}, v_q)$, replace $d(v_{i_j}, v_q) + d(v_p, v_q)$ in (4.27) with $d(v_{i_j}, v_q)$, then

$$f_2' \geq f_2'' = \sum_{j=r+1}^{n} \left[\frac{d(v_{i_j}, v_p)K_{i_j}}{d(v_{i_j}, v_p) + K_{i_j}} \right.$$
$$\left. - \frac{d(v_p, y_0)K_{i_j}^2}{(d(v_{i_j}, v_q) + d(v_p, v_q) - d(v_p, y_0) + K_{i_j})(d(v_{i_j}, v_q) + K_{i_j})} \right].$$

Hence,

$$f \geq f_1 + f_2'' \tag{4.28}$$

If (4.28) is rearranged, then

$$f \geq \sum_{j=1}^{n} \frac{d(v_{i_j}, v_p)K_{i_j}}{d(v_{i_j}, v_p) + K_{i_j}} \tag{4.29}$$
$$+ d(v_p, y_0) \left[\sum_{j=1}^{r} \frac{K_{i_j}^2}{(d(v_{i_j}, v_p) + d(v_p, y_0) + K_{i_j})(d(v_{i_j}, v_p) + K_{i_j})} \right. \tag{4.30}$$
$$\left. - \sum_{j=r+1}^{n} \frac{K_{i_j}^2}{(d(v_{i_j}, v_q) + d(v_p, v_q) - d(v_p, y_0) + K_{i_j})(d(v_{i_j}, v_q) + K_{i_j})} \right]. \tag{4.31}$$

Summations in (4.30) and (4.31) are equal to (4.32) and (4.33), respectively.

$$\sum_{j=1}^{r} p_{iy_0} p_{iv_p} \tag{4.32}$$
$$\sum_{j=r+1}^{n} p_{iy_0} p_{iv_q} \tag{4.33}$$

If (4.34) is satisfied, (4.35) will hold true. That is, v_p is a better location than y_0 for x_c.

$$\sum_{j=1}^{r} p_{iy_0} p_{iv_p} \geq \sum_{j=r+1}^{n} p_{iy_0} p_{iv_q} \tag{4.34}$$
$$\sum_{j=1}^{n} \frac{d(v_{i_j}, y_0)K_{i_j}}{d(v_{i_j}, y_0) + K_{i_j}} \geq \sum_{j=1}^{n} \frac{d(v_{i_j}, v_p)K_{i_j}}{d(v_{i_j}, v_p) + K_{i_j}} \tag{4.35}$$

If (4.34) is not satisfied, then we have

$$\sum_{j=1}^{r} p_{iy_0} p_{iv_p} < \sum_{j=r+1}^{n} p_{iy_0} p_{iv_q} \tag{4.36}$$

Clearly, (4.35) is not guaranteed in this case.

Again, let

$$f = f_1 + f_2$$

Add and subtract $d(v_p, v_q)$ both numerators and denominators of each term of summation of f_1 in (4.23), then

$$f_1 = \sum_{j=1}^{r} \frac{(d(v_{i_j}, v_p) + d(v_p, y_0) + d(v_p, v_q) - d(v_p, v_q))K_{i_j}}{d(v_{i_j}, v_p) + d(v_p, y_0) + d(v_p, v_q) - d(v_p, v_q) + K_{i_j}}, \tag{4.37}$$

Multiply both the numerator and denominator of each term of the summation of f_1 in (4.37) by $d(v_{i_j}, v_p) + d(v_p, v_q) + K_{i_j}$, then

$$f_1 = \sum_{j=1}^{r} \left[\frac{(d(v_{i_j}, v_p) + d(v_p, v_q))K_{i_j}}{d(v_{i_j}, v_p) + d(v_p, v_q) + K_{i_j}} \right. \\ \left. - \frac{(d(v_p, v_q) - d(v_p, y_0))K_{i_j}^2}{(d(v_{i_j}, v_p) + d(v_p, v_q) - d(v_p, v_q) + d(v_p, y_0) + K_{i_j})(d(v_{i_j}, v_p) + d(v_p, v_q) + K_{i_j})} \right].$$

Cancelling the $d(v_p, v_q) - d(v_p, v_q)$ expression, we have

$$f_1 = \sum_{j=1}^{r} \left[\frac{(d(v_{i_j}, v_p) + d(v_p, v_q))K_{i_j}}{d(v_{i_j}, v_p) + d(v_p, v_q) + K_{i_j}} \right. \tag{4.38}$$

$$\left. - \frac{(d(v_p, v_q) - d(v_p, y_0))K_{i_j}^2}{(d(v_{i_j}, v_p) + d(v_p, y_0) + K_{i_j})(d(v_{i_j}, v_p) + d(v_p, v_q) + K_{i_j})} \right]. \tag{4.39}$$

Similarly, multiply both the numerator and denominator of each term of the summation of f_2 in (4.24) by $(d(v_{i_j}, v_q) + K_{i_j})$, then

$$f_2 = \sum_{j=r+1}^{n} \left[\frac{d(v_{i_j}, v_q)K_{i_j}}{d(v_{i_j}, v_q) + K_{i_j}} \right. \\ \left. + \frac{((d(v_p, v_q) - d(v_p, y_0))K_{i_j}^2}{(d(v_{i_j}, v_q) + d(v_p, v_q) - d(v_p, y_0) + K_{i_j})(d(v_{i_j}, v_q) + K_{i_j})} \right].$$

By triangle inequality, we have $d(v_{i_j}, v_p) + d(v_p, v_q) \geq d(v_{i_j}, v_q)$. Then

$$\sum_{j=1}^{r} \frac{(d(v_{i_j}, v_p) + d(v_p, v_q))K_{i_j}}{d(v_{i_j}, v_p) + d(v_p, v_q) + K_{i_j}} \geq \sum_{j=1}^{r} \frac{d(v_{i_j}, v_q)K_{i_j}}{d(v_{i_j}, v_q) + K_{i_j}}. \tag{4.40}$$

Substitute right-hand side of (4.40) in (4.38), we have

$$f_1 \geq f_1' = \sum_{j=1}^{r} \left[\frac{d(v_{i_j}, v_q)K_{i_j}}{d(v_{i_j}, v_q) + K_{i_j}} \right. \\ \left. - \frac{(d(v_p, v_q) - d(v_p, y_0))K_{i_j}^2}{(d(v_{i_j}, v_p) + d(v_p, y_0) + K_{i_j})(d(v_{i_j}, v_p) + d(v_p, v_q) + K_{i_j})} \right]. \tag{4.41}$$

Since $d(v_{i_j}, v_p) + d(v_p, v_q) \geq d(v_{i_j}, v_p)$, replace $d(v_{i_j}, v_p) + d(v_p, v_q)$ in (4.41) with $d(v_{i_j}, v_p)$, then

$$f_1' \geq f_1'' = \sum_{j=1}^{r} \left[\frac{d(v_{i_j}, v_q)K_{i_j}}{d(v_{i_j}, v_q) + K_{i_j}} - \frac{(d(v_p, v_q) - d(v_p, y_0))K_{i_j}^2}{(d(v_{i_j}, v_p) + d(v_p, y_0) + K_{i_j})(d(v_{i_j}, v_p) + K_{i_j})} \right].$$

Hence,

$$z \geq f_1'' + f_2 \tag{4.42}$$

If (4.42) is rearranged, then

$$\begin{aligned} z \geq & \sum_{j=1}^{n} \frac{d(v_{i_j}, v_q)K_{i_j}}{d(v_{i_j}, v_q) + K_{i_j}} \\ & + (d(v_p, v_q) - d(v_p, y_0)) && (4.43) \\ & * \left[\sum_{j=r+1}^{n} \frac{K_{i_j}^2}{(d(v_{i_j}, v_q) + d(v_p, v_q) - d(v_p, y_0) + K_{i_j})(d(v_{i_j}, v_q) + K_{i_j})} \right. && (4.44) \\ & \left. - \sum_{j=1}^{r} \frac{K_{i_j}^2}{(d(v_{i_j}, v_p) + d(v_p, y_0) + K_{i_j})(d(v_{i_j}, v_p) + K_{i_j})} \right]. && (4.45) \end{aligned}$$

Summations in (4.44) and (4.45) are equal to (4.32) and (4.33), respectively. If (4.36) is satisfied, (4.46) will hold true. v_q is a better location than y_0 for x_c.

$$\sum_{i=1}^{n} \frac{d(v_i, y_0)K_i}{d(v_{i_j}, y_0) + K_i} \geq \sum_{i=1}^{n} \frac{d(v_i, v_q)K_i}{d(v_{i_j}, v_q) + K_i} \tag{4.46}$$

As shown above, when (4.34) is satisfied. v_p is a better location than y_0 for x_c. Otherwise, (when (4.36) is satisfied), v_q is a better location than y_0 for x_c. As a result, the center x_c will always be located on a vertex. □

This proof supports Levy's proof that in the case of concavity, the center will be located on $\mathbf{V}$. This result could be generalized such that all the cluster centers are located on vertices in optimal solution.

Theorem 4.2.3. *For every cluster center* $\{x_c, x_2, ..., x_p\} \in G$*, there exists* $\{v_{m_1}, v_{m_2}, ..., v_{m_p}\} \in V$ *such that*

$$\sum_{k=1}^{P} \sum_{i=1}^{n} p_{ik}^2 d(v_i, x_k) \geq \sum_{k=1}^{P} \sum_{i=1}^{n} p_{ik}^2 d(v_i, v_k)$$

Proof. To prove this theorem, we will change location of a center i (x_i) by fixing the other centers ($x_k, k \neq i$). In each location change, membership scores will change. Initial membership score will be denoted as $p_{ik}^{(0)}$. Updated membership scores resulting from changing location of cluster center k will be denoted as $p_{ik}^{(k)}$.

Let x_c be the cluster center to be changed, keeping the others fixed. In previous theorem, we have shown that

$$\sum_{k=1}^{P}\sum_{i=1}^{n} p_{ik}^{(0)^2} d(v_i, x_k) \quad \geq \quad \sum_{i=1}^{n} p_{ik}^{(1)^2} d(v_i, v_{m_1}) \; + \; \sum_{k=2}^{P}\sum_{i=1}^{n} p_{ik}^{(1)^2} d(v_i, x_k). \quad (4.47)$$

Now, let x_2 be the cluster center to be changed, keeping the others fixed in their last locations. Again, we have

$$\sum_{i=1}^{n} p_{ik}^{(1)^2} d(v_i, v_{m_1}) + \sum_{k=2}^{P}\sum_{i=1}^{n} p_{ik}^{(1)^2} d(v_i, x_k) \geq$$
$$\sum_{k=1}^{2}\sum_{i=1}^{n} p_{ik}^{(2)^2} d(v_i, v_{m_k}) + \sum_{k=3}^{P}\sum_{i=1}^{n} p_{ik}^{(2)^2} d(v_i, x_k). \quad (4.48)$$

Perform this process with each x_k as the center location to be changed, as the last expression, we have

$$\sum_{k=1}^{p-1}\sum_{i=1}^{n} p_{ik}^{(p-1)^2} d(v_i, v_{m_k}) + \sum_{i=1}^{n} p_{ik}^{(p-1)^2} d(v_i, x_p) \geq \sum_{k=1}^{p}\sum_{i=1}^{n} p_{ik}^{(p)^2} d(v_i, v_{m_k}). \quad (4.49)$$

Combine (4.47), (4.48) and (4.49), we have

$$\sum_{k=1}^{P}\sum_{i=1}^{n} p_{ik}^{(0)^2} d(v_i, x_k) \geq \sum_{k=1}^{P}\sum_{i=1}^{n} p_{ik}^{(p)^2} d(v_i, v_{m_k}),$$

which implies that as center locations, $v_{m_1}, ..., v_{m_k}$ lead to a better solution than $x_c, ..., x_k$. □

As a result, it has been shown that in PD-Clustering Problem on Networks, the optimal solution will always be located on vertices. Therefore, one would not lose from objective function if they restrict center locations as $\mathbf{V}$ instead of $\mathbf{G}$.

4.2.2 Fuzzy Clustering (FC) Problem on Networks

Fuzzy Clustering Problem on Networks is defined as

$$\text{minimize} \quad f(\mathbf{X}) = \sum_{i=1}^{n}\sum_{k=1}^{p} p_{ik}^{m} d(v_i, x_k)^2 \tag{4.50}$$

subject to

$$\sum_{k=1}^{p} p_{ik} = 1 \quad \forall\, i = 1, ..., n,$$

$$p_{ik} \geq 0 \quad \forall\, i = 1, ..., n,\ k = 1, ..., p,$$

$$x_k \in G \quad \forall\, k = 1, ..., p,$$

where x_k is the location of cluster center k and p_{ik} is the membership value of v_i to cluster k. m is called *fuzzifier*, or *fuzziness index*. It determines the level of fuzziness in memberships. If $m = 1$, the problem becomes a hard assignment problem - to be more precise, SSC problem. As m gets larger, all membership values converge to $1/p$. For each vertex, summation of memberships to all clusters must be equal to 1. Derived by Bezdek et.al. in [12] with the use of Lagrangian, keeping all x_k fixed, membership function is

$$p_{ik}^{*} = \frac{1}{\sum_{l=1}^{p} \left(\frac{d(v_i,x_k)}{d(v_i,x_l)}\right)^{\frac{2}{(m-1)}}}. \tag{4.51}$$

When this problem has been investigated, it has been observed that the optimal center locations are could be on anywhere on the $\mathbf{G}$. In this subsection, this property will be analyzed.

Fuzzy Clustering with a Single Cluster

As in PD-Clustering, if there is a single cluster, all vertices will have a membership equal to 1. FC with 1 cluster differs from PD-Clustering in that (4.50), becomes sum of squared distances. Therefore, the problem will demonstrate characteristics of SSC problem with 1 cluster. In a $\mathbf{G}$ with one cluster x_c moving along the edge (v_p, v_q), f_i, the objective function component of v_i, will be as in Figure 4.6. f_i is a second degree polynomial function increasing with $d(v_i, x_k)$. f_i is convex or piecewise on an edge. This piecewiseness occur at arc bottleneck points. Each piece of f_i is convex and increasing with distance. Because of the convexity, f which is summation of f_i functions is also convex. But it is not monotone; therefore, it may contain a local minimum along an edge. As a result, there could be an optimal center location located on interior point of an edge. Based on this observation, we can conclude that the following theorem holds.

Theorem 4.2.4. *Let* $\mathbf{V}^*$ *be a set of p vertices* $\{v_1^*, v_2^*, ..., v_p^*\}$ *which is the optimal solution among all possible* $\mathbf{V}$ *sets. In Fuzzy Clustering Problem on networks with a single cluster, there exists a subset* $\mathbf{X}^* \in \mathbf{G}$ *containing centers located on edges such that it has an objective function value lower than* $\mathbf{V}^*$*.*

Fuzzy Clustering with p Clusters

In this subsection, objective function of FC Problem with p clusters will be analyzed and structural properties will be investigated.

For the sake of simplicity, suppose we have two clusters. If (4.51) is evaluated for this case, membership value of vertex i will be

$$p_{i1} = \frac{d(v_i, x_2)^{\frac{2}{(m-1)}}}{d(v_i, x_c)^{\frac{2}{(m-1)}} + d(v_i, x_2)^{\frac{2}{(m-1)}}}, \quad p_{i2} = \frac{d(v_i, x_c)^{\frac{2}{(m-1)}}}{d(v_i, x_c)^{\frac{2}{(m-1)}} + d(v_i, x_2)^{\frac{2}{(m-1)}}}. \tag{4.52}$$

For a graph $\mathbf{G}$ with two clusters as in Figure 4.2, keeping x_2 fixed and moving x_c on the edge (v_p, v_q), change in the membership functions p_{i1} and p_{i2} has been visualized in Figure 4.10 with fuzziness index m value of 2. As in PD-Clustering, both memberships are affected by the location change of x_c. In (a), (b) and (c), as x_c moves towards arc bottleneck point b_i, p_{i1} decreases due to the increase in distance. As p_{i1} decreases, p_{i2} increases. (a) illustrates the case $d(v_i, x_1)$ is less than $d(v_i, x_2)$; therefore, p_{i2} is less than p_{i1}. (c) is the case $d(v_i, x_1)$ is less than $d(v_i, x_2)$; as a result, p_{i1} is less than p_{i2}. In (b), if $x_c \in [v_p, a_i^p]$, p_{i1} is greater than p_{i2}. If $x_c \in [a_i^p, a_i^q]$, p_{i2} is greater than p_{i1}. In the last interval which is $x_c \in [a_i^q, v_q]$, $d(v_i, x_1)$ is less than $d(v_i, x_2)$. Hence, p_{i1} is greater than p_{i2}. In (b), assignment bottleneck points could be observed as the points where p_{i1} =p_{i2}. Figure 4.11 is drawn with fuzziness index $m = 20$ to illustrate effect of increase in m in membership function. As could be observed, it does not change the behavior of the membership function. However, even at the points where $d(v_i, x_1)$ values have the maximum difference, memberships p_{i1} and p_{i2} are very close to each other compared to the case of m=2. The effect of increase in m is increase in the fuzziness of the memberships.

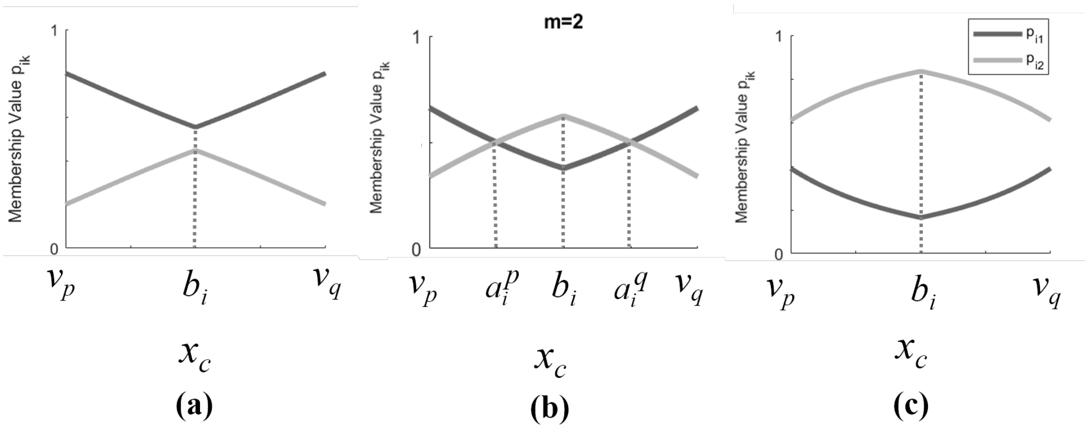

Figure 4.10: Membership function p_{ik} of FC with 2 clusters when m=2

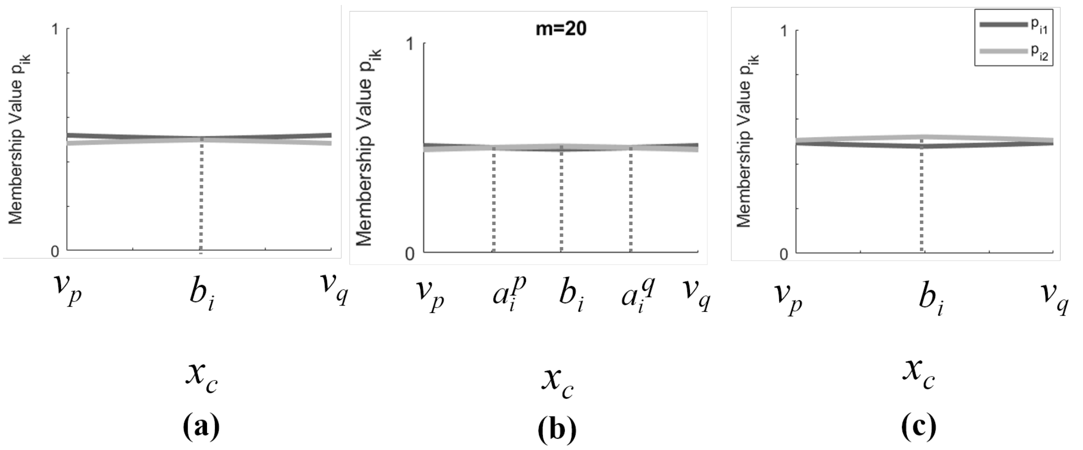

Figure 4.11: Membership function p_{ik} of FC with 2 clusters when m=20

If (4.17) is substituted in (4.50), the objective function will be

$$f(\mathbf{X}) = \sum_{i=1}^{n} \frac{d(v_i, x_c)^2 d(v_i, x_2)^2}{\left(d(v_i, x_c)^{\frac{2}{(m-1)}} + d(v_i, x_2)^{\frac{2}{(m-1)}}\right)^{m-1}}.$$

For three clusters, the membership values are

$$p_{i1} = \frac{(d(v_i,x_2)d(v_i,x_3))^{\frac{2}{(m-1)}}}{(d(v_i,x_c)d(v_i,x_2))^{\frac{2}{(m-1)}} + (d(v_i,x_2)d(v_i,x_3))^{\frac{2}{(m-1)}} + (d(v_i,x_c)d(v_i,x_3))^{\frac{2}{(m-1)}}},$$
$$p_{i2} = \frac{(d(v_i,x_c)d(v_i,x_3))^{\frac{2}{(m-1)}}}{(d(v_i,x_c)d(v_i,x_2))^{\frac{2}{(m-1)}} + (d(v_i,x_2)d(v_i,x_3))^{\frac{2}{(m-1)}} + (d(v_i,x_c)d(v_i,x_3))^{\frac{2}{(m-1)}}},$$
$$p_{i3} = \frac{(d(v_i,x_c)d(v_i,x_2))^{\frac{2}{(m-1)}}}{(d(v_i,x_c)d(v_i,x_2))^{\frac{2}{(m-1)}} + (d(v_i,x_2)d(v_i,x_3))^{\frac{2}{(m-1)}} + (d(v_i,x_c)d(v_i,x_3))^{\frac{2}{(m-1)}}}.$$

With the same manner, the objective function could be written as

$$f(\mathbf{X}) = \sum_{i=1}^{n} \frac{(d(v_i,x_c)d(v_i,x_2)d(v_i,x_3))^2}{\left((d(v_i,x_c)d(v_i,x_2))^{\frac{2}{(m-1)}} + (d(v_i,x_2)d(v_i,x_3))^{\frac{2}{(m-1)}} + (d(v_i,x_c)d(v_i,x_3))^{\frac{2}{(m-1)}}\right)^{m-1}}.$$

In the version of the problem with p clusters, the objective function is

$$f = \sum_{i=1}^{n} \frac{\prod_{k\in K} d(v_i,x_k)^2}{\left(\sum_{k\in K}\prod_{l\neq k} d(v_i,x_l)^{\frac{2}{(m-1)}}\right)^{m-1}} \tag{4.53}$$

For the sake of simplicity, we assume that $m = 2$. Since we assume that location of x_k is fixed for $k = 1, ..., p - \{c\}$, we could separate constant components of each vertex as K_i and write the objective function (4.53) as in (4.54).

$$K_i = \frac{\prod_{k=2}^{p} d(v_i,x_k)^2}{\sum_{k=2}^{p}\prod_{l\neq k} d(v_i,x_l)^2} \rightarrow \qquad f(x_c) = \sum_{i=1}^{n} \frac{d(v_i,x_c)^2 K_i}{d(v_i,x_c)^2 + K_i} \tag{4.54}$$

f_i, contribution of vertex v_i to the function (4.19), is continuous and twice differentiable. First and second order derivatives are

$$\begin{aligned} \frac{df_i}{dx_c} &= \frac{2K_i^2}{(K_i + d(v_i,x_c)^2)^2} \\ \frac{d^2 f_i}{dx_c^2} &= \frac{(2K_i^2)(K_i - 3d(v_i,x_c)^2)}{(K_i + d(v_i,x_c)^2)^3}. \end{aligned} \tag{4.55}$$

With the second derivative test, f_i is

- Convex if $\frac{d^2 f_i}{dx_c^2} \geq 0$, that is, $d(v_i,x_c) \leq \sqrt{\frac{K_i}{3}}$,

- Concave if $\frac{d^2 f_i}{dx_c^2} \geq 0$, that is, $d(v_i, x_c) \geq \sqrt{\frac{K_i}{3}}$.

Theorem 4.2.5. *In Fuzzy Clustering Problem with p clusters, given a solution* $\mathbf{X}$ $= \{x_1, x_2, ..., x_p\}$*, if* $x_c \in \mathbf{X}$ *is moved along a given edge keeping the centers* $\mathbf{X} - x_c$ *fixed, there could be a location on the given edge that minimizes the objective function* (4.53).

Proof. Let

$$c_i = \prod_{k=2}^{p} d(v_i, x_k)$$
$$t_i = \sum_{k=2}^{p} \prod_{j \in K, j \neq k} d(v_i, x_j)^{\frac{2}{m-1}}.$$

If we fix locations of x_k, $k \in K - \{1\}$ and separate fixed components of f_i from variable components by using c_i and t_i, the objective function is (4.56).

$$f_i = \frac{c_i^2 d(v_i, x_c)^2}{\left(x^{\frac{2}{m-1}} t_i + c_i^{\frac{2}{m-1}}\right)^{m-1}} \tag{4.56}$$

If we calculate first and second order derivatives for f_i with any $m > 1$, we have

$$\frac{df_i}{dx_c} = \frac{(2c_i^2 d(v_i, x_c))}{(c_i + t_i d(v_i, x_c)^{\frac{2}{m-1}})^m} \tag{4.57}$$

$$\frac{d^2 f_i}{dx_c^2} = \frac{(2c_i^2)\left(c_i(m-1) - t_i(m+1) d(v_i, x_c)^{\frac{2}{m-1}}\right)}{\left(\left(c_i + t_i d(v_i, x_c)^{\frac{2}{m-1}}\right)(m-1)\right)^{m+1}} \tag{4.58}$$

By the second derivative test, f_i is

- Convex if $\frac{d^2 f_i}{dx_c^2} \geq 0$, that is, $d(v_i, x_c) \geq \sqrt[\frac{2}{m-1}]{\frac{c_i(m-1)}{t_i(m+1)}}$,
- Concave if $\frac{d^2 f_i}{dx_c^2} \geq 0$, that is, $d(v_i, x_c) < \sqrt[\frac{2}{m-1}]{\frac{c_i(m-1)}{t_i(m+1)}}$.

As a result, f_i is a nonconvex function of $d(v_i, x_k)$, and increasing with distance. The objective function could be illustrated as in Figure 4.12. As in PD-Clustering, f_i is piecewise at arc bottleneck points. And each piece of f_i is nonconvex according to the sign of second derivative (4.58). Since summation of nonconvex functions are nonconvex, f, summation of $f_i \, \forall \, i = 1, ..., n$, is nonconvex. Since f is not monotone, local minimum could be found at a point where second derivative is positive and first derivative is zero.

Let **G** be a graph with *p* clusters. Keeping *p-1* clusters fixed and moving one cluster center (let us say x_c) along the edge (v_p, v_q), f_i function could be observed as given in Figure 4.9. In (a) and (c), the shortest path from v_i to x_c passes from v_p and v_q, respectively. In (b), when $x_c \in [v_p, b_i]$, f_i increases. When $x_c \in [b_i, v_q]$, f_i decreases with the decreasing distance. In (b), f_i is piecewise concave. Different from hard assignment problems, piecewiseness occurs at only arc bottleneck points only. Since each f_i is concave or piecewise concave, f, summation of $f_i \, \forall \, i = 1, ..., n$, is also concave.

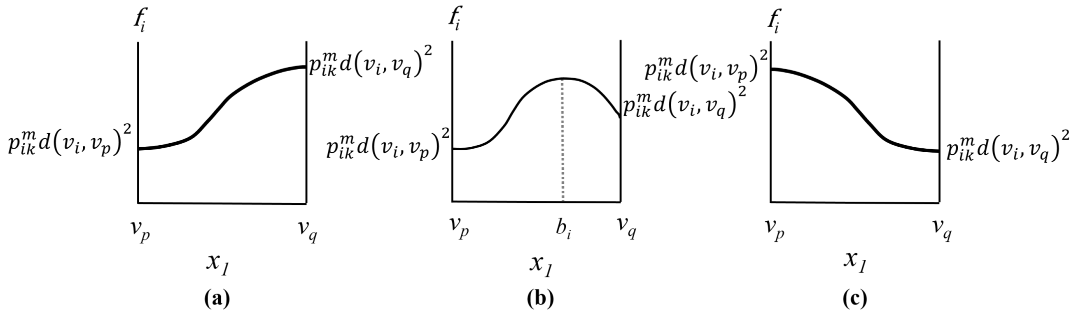

Figure 4.12: Objective function component for v_i (denoted as f_i) in FC problem with p clusters when x_c is moved along the edge (v_p, v_q) and x_k is the second closest cluster center to v_i

Hence, in Fuzzy Clustering Problem, one may find an optimal solution that contains cluster centers located at edges.

□

CHAPTER 5

SOLUTION APPROACHES

In this chapter, solution approaches for Hard Assignment Problems (P-Median and SSC) and Soft Assignment Problems (PD-Clustering and Fuzzy Clustering (FC)) will be discussed. Two Hybrid Genetic Algorithms have been developed to solve these problems:

- Node Based Hybrid Genetic Algorithm (HGA-N),
- Edge Based Hybrid Genetic Algorithm (HGA-E).

These two solution approaches have been developed based on the same principles. Main difference occurs in the local search procedure (explained in detail in §5.3) and solution space. While the solution space that HGA-N searches for is restricted to $\mathbf{V}$, solution space in HGA-E is the entire network $\mathbf{G} = (\mathbf{E}, \mathbf{V})$.

This chapter has been organized as follows. In §5.1, Genetic Algorithm and fundamentals of it will be discussed. In §5.2, general framework of the Local Search procedure that is implemented will be provided. In §5.3, Hybrid Genetic Algorithm structure will be presented. Then, two versions of HGA which are HGA-N and HGA-E will be discussed in detail.

5.1 Genetic Algorithm

According to evolutionary theory in biology, at a time t, a population exists with different individuals. As time passes, new individuals are born, and they take some characteristics from their parents. Due to *natural selection* and *survival of the fittest*, better individuals survive. With this way, the population undergoes evolution, and better individuals appear. As a metaheuristic approach inspired from biology, Genetic Algorithm (GA) works in the same manner. GA was first proposed by Holland in 1975 [14]. Population corresponds to the set of solutions, which are expressed with chromosomes. Parents are selected from the population and offsprings are generated by crossover operation. Offsprings are mutated with a specified probability. Then, by checking the quality of the offsprings (it is performed with fitness value, which generally corresponds to the objective function), a new generation is created. This procedure is repeated iteratively until a termination condition is satisfied. In GA (as in any heuristic), as Lozano and Martinez [30] stated, both exploring solution space and exploiting the regions that are likely to contain optimal solution is important, and these two sometimes-conflicting goals must be balanced.

Typically, parent selection and generation replacement operators promote intensification while crossover and mutation operators promote diversification. In order not to get stuck at a local optimal solution, early convergence situation must be handled with the help of operators and termination condition. Moreover, slow convergence issue must be handled to reduce computational effort by fine-tuning the level of diversification in GA.

5.2 Local Search (LS) Procedure

In our approach, we use a Local Search (LS) procedure inside the GA. The problem of getting stuck at local optima is eliminated with the help of GA, whereas LS aims to help GA converge faster to good solutions. A sample trace for LS is given in Figure 5.1 (a)-(d). The procedure starts with a cluster center. In the figure, it is denoted as x_k. As could be seen in (a), x_k is located on the edge between vertices 2 and 5. The first step is to search for the best location of x_k on the edge (2,5). Then, x_k is updated and edge (2,5) is added to the set of visited edges. Meanwhile, vertices 2 and 5 are added to a reference list since their adjacent edges will be visited in the next iteration. As in (b), edges (2,6), (2,3), (2,7), (5,8) and (5,9) are found as the edges to be visited. Locations with the minimum fitness function value on each edge is found. If there is an improvement in the fitness function, x_k is located to the best location, which is on edge (2,3) as in (c) in the figure. Then, set of visited edges are updated again (the edges with the grey color are visited) and reference list is cleared and populated with the end vertices 3 and 7, and end vertices of other visited edges with a fitness function value equal to the current fitness function value (if any). Then, edges (3,4), (3,1), (3,10) and (3,7) are selected as the edges to be visited next. If there is an improvement, x_k is updated again. In (d), since there is a better solution, x_k is moved to the edge (3,7). This procedure continues for a cluster center until there is no improvement in the fitness function value. When there is no improvement, other cluster centers undergo the same procedure.

In our LS algorithm, first, centers are shuffled so that the order in the chromosome representation is not important. Then, the number of cluster centers to be improved is calculated according to a parameter β, which is the portion of cluster centers to be improved in a solution. For $\lceil \beta * p \rceil$ centers, the following procedure is executed iteratively. For a center x_k, alternative locations that are adjacent (connected) to x_k is found and objective (fitness) function values are calculated. If there is a better alternative than the current location, x_k is updated, and unvisited adjacent locations of the new x_k is searched in the next iteration. If there is not a better alternative than the current location x_k, adjacents of x_{k+1} are sought in the next iteration. Given a solution with p centers, flowchart of LS Algorithm is presented in Figure 5.2. This procedure continues until the number of cluster centers sought reached $\lceil \beta * p \rceil$. In the algorithm, we have the parameter β since we discovered that if we try to improve all cluster centers, the last iterations do not improve the fitness function value significantly, while they consume time. An illustration for this concept is given in Figure 5.3, in which x-axis refers to cluster centers and y-axis refers to fitness value as LS procedure continues. In this figure, a solution with 20 clusters is taken. The first center improved is cluster center 8. Then, cluster center 5 is improved. If the improvement is done for all cluster centers, the fitness value decreases to 3198, which is a 18% improvement on the initial solution fitness of 3917. If the procedure is terminated at cluster center 12 which corresponds to $\beta = 0.7$, the objective function decreases to 3343, which is a 14% improvement on the solution. So, the local search for remaining centers do not improve the fitness value significantly. The vertical dashed line on the figure illustrates the point for $\beta = 0.7$.

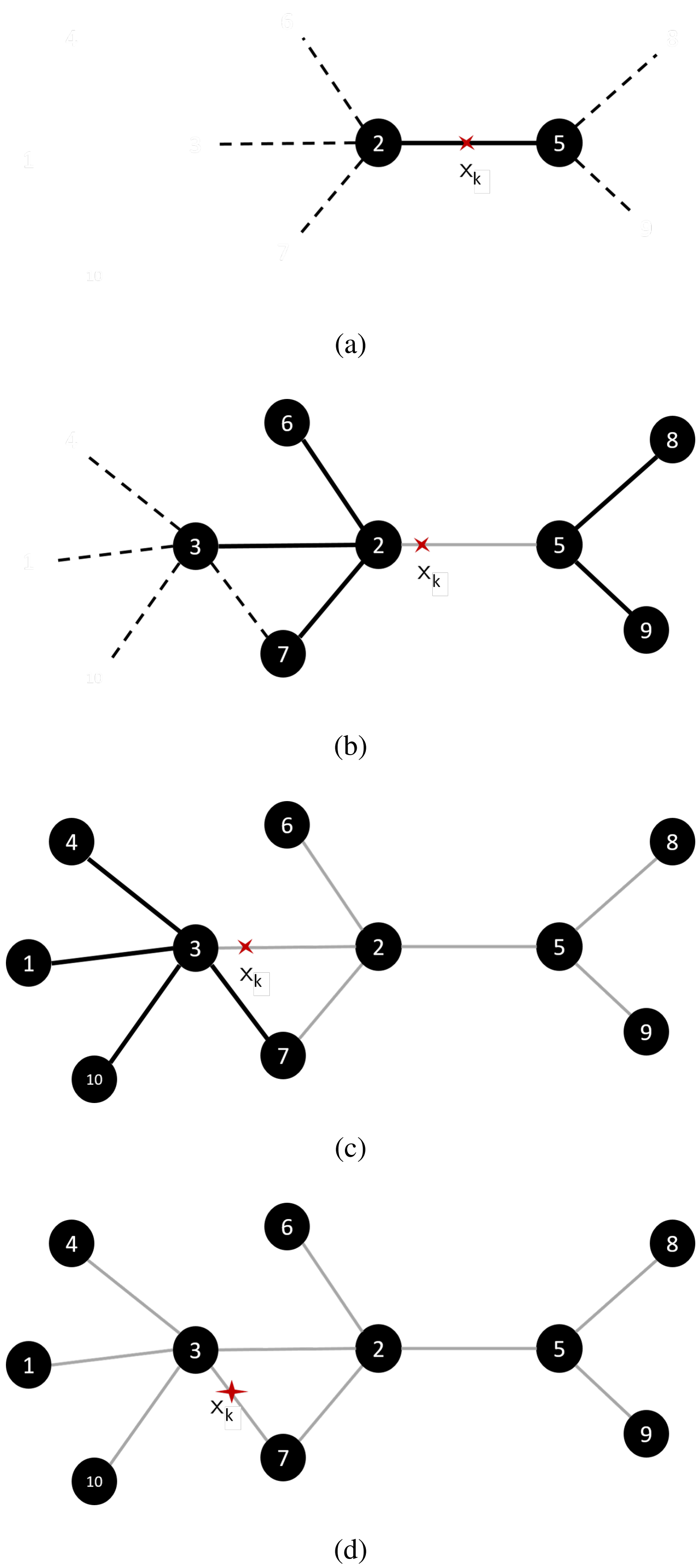

Figure 5.1: A local search example for cluster center x_k

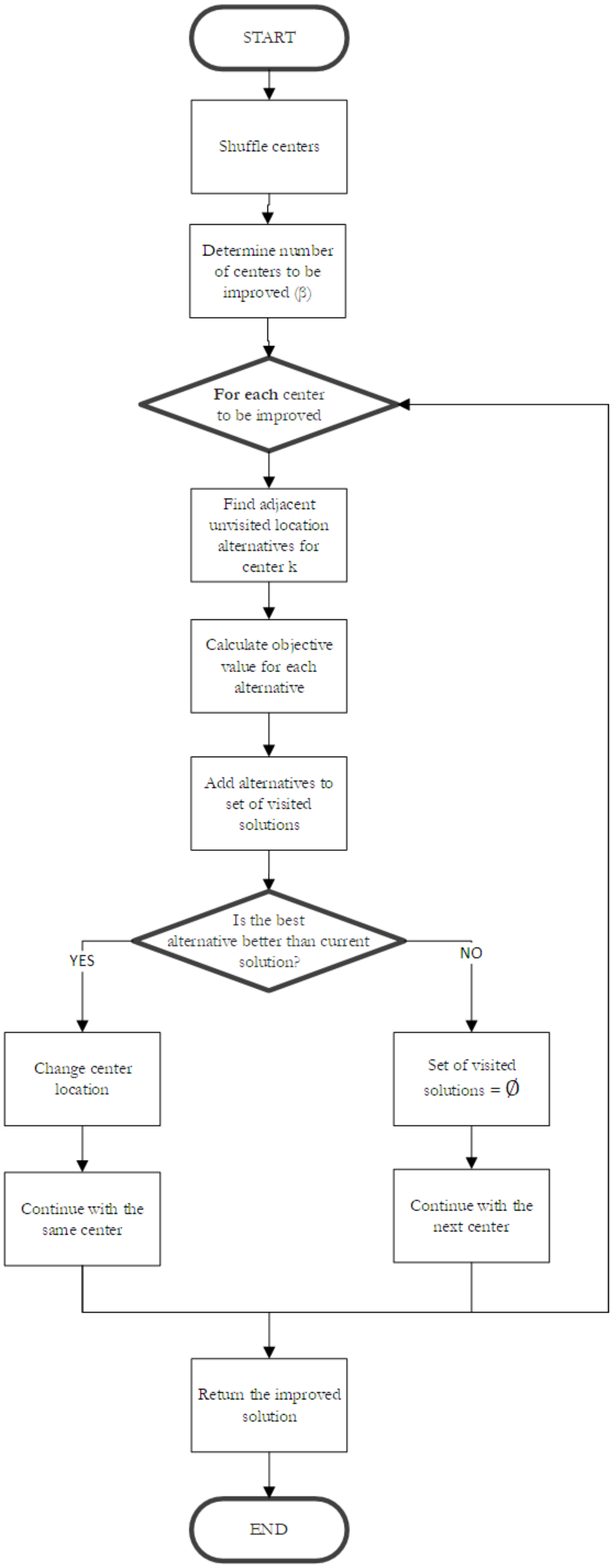

Figure 5.2: Flowchart of the LS algorithm

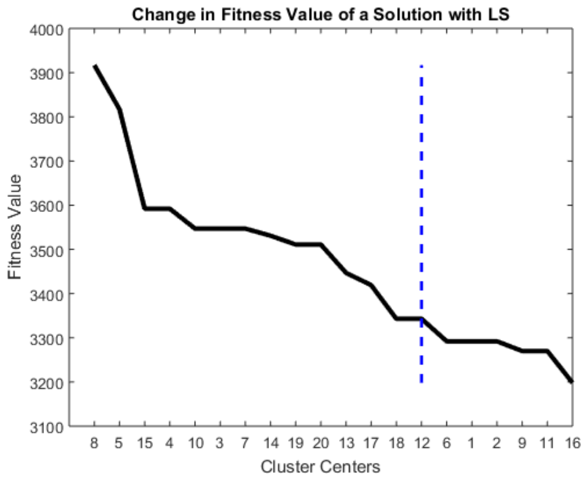

Figure 5.3: Effect of the LS algorithm on a solution

In short, the LS procedure searches for a better solution on adjacent locations on the graph $\mathbf{G}$. With the parameter β, an early stopping is imposed in order to eliminate insignificant moves and improve computational efficiency. This procedure has been implemented in both HGA-N and HGA-E with differences stem from the solution space difference, which will be discussed in Subsections 5.3.1 and 5.3.2 in further detail.

5.3 Hybrid Genetic Algorithm (HGA)

GA is one of the most well-known metaheuristics that has been proven to find promising results to a wide range of problems. There are GA implementations in the literature that has good performance such as [27] for P-Median Problem. In this study, we chose to apply a variant of GA to the problems on hand. In [31], it is stated that LS procedures could enhance the performance of GAs. We developed an approach by combining GA with our LS, which we named as HGA. HGA provides a unified solution framework, and it can solve different problem types by modifying it for each problem type.

In HGA, in each generation, the fittest offspring is selected and local search operation has been carried out. Mutation operation is not designed because it has been observed that it increases runtime while it does not improve performance significantly. The framework we used in our proposed GA has been given in Figure 5.4. In HGA, with the help of crossover operation, diversification has been promoted, and intensification is promoted by local search operation. Generation replacement has been designed so that it balances diversification and intensification. Also, it is worth to note that in HGA, objective function of the corresponding problem has been used as fitness function. Using this general framework, HGA is modified for the problem types and their properties. Further details will be discussed in Subsections 5.3.1 and 5.3.2.

5.3.1 Node Based Hybrid Genetic Algorithm (HGA-N)

This algorithm has been developed for P-Median Problem and PD-Clustering Problem. In these two problems, it is shown that the optimal cluster centers are always located on vertices. Regarding this condition, a tailored version of our approach is preferred by restricting the solution space to $\mathbf{V}$ for finding the solution more efficiently. This version is named as HGA-N.

Chromosome Representation

In HGA-N, a node based approach is followed. Additionally, number of clusters p is a predefined value. The chromosome is an array containing p values each representing vertex index of each center, similar to the representation scheme which was used by Alp et al. in [27]. Chromosome representation is visualized in Figure 5.5. With this representation, it is guaranteed that number of clusters are exactly p. Since empty clusters are not encountered at optimality, empty cluster situation is not checked.

Initial Population Generation

In this stage, we generate each individual randomly with randomly selected vertices. During the initialization, it is guaranteed that no duplicate solutions are generated until the population size exceeds the maximum number of different individuals that exist in the solution space, $u_{max} = \binom{n}{k}$, where n is the number of vertices and p is the number of clusters. Pseudocode of the procedure has been provided below as

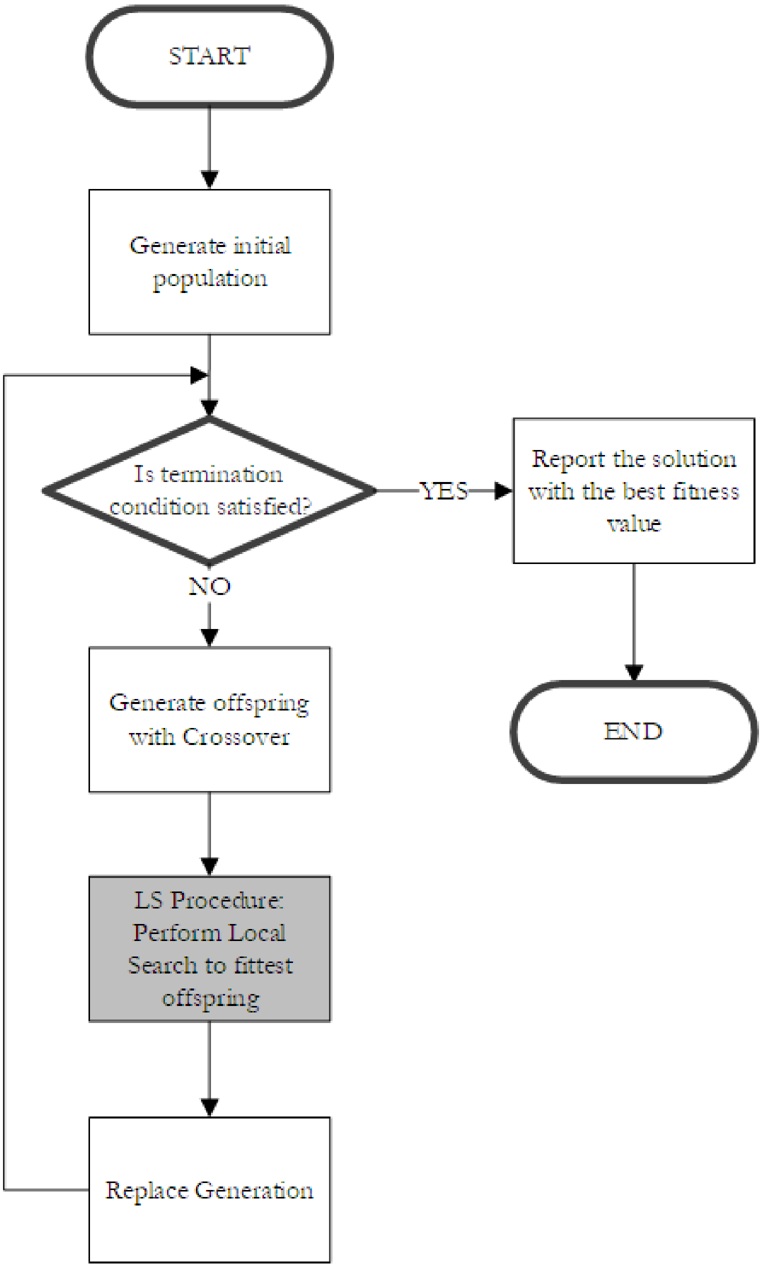

Figure 5.4: Flowchart of the HGA

x_1	x_2	...	x_p

Figure 5.5: Chromosome representation in HGA-N algorithm

Algorithm 4. If population size is determined with the function given, u_{max} is not exceeded. However, for problems with smaller size, if population size exceeds u_{max}, duplicate solutions could be added to population.

Algorithm 4 Population Generation in HGA-N

```
Output: Population
while PopulationSize is not reached do
    Generate an individual randomly.
    if The new individual is not a duplicate then
        Add the new individual to Population.
    else if Number of generated individuals are greater than or equal to u_max then
        Add the new individual to Population.
    else
        Continue
    end if
end while
```

The population size used in HGA-N depends on the size of the problem instance. Population size is calculated as

$$PopulationSize = \max\{10, \lceil \sqrt[3]{n} * \ln(u_{max}) \rceil\}, \tag{5.1}$$

where u_{max} is the maximum number of unique solutions. So, in (5.1), population size depends on the instance size (number of vertices) and size of the solution space (maximum number of unique solutions). Change of population size with the instance size is illustrated in Figure 5.6.

Crossover

Crossover starts with selecting two parents for reproduction. The selection procedure could be completely random, or *elitist*, that is, individuals with better fitness function values could be selected with a higher probability. After selection, offsprings are generated. Depending on the design of GA, there could be a number of offsprings (usually, two offsprings are generated from two parents).

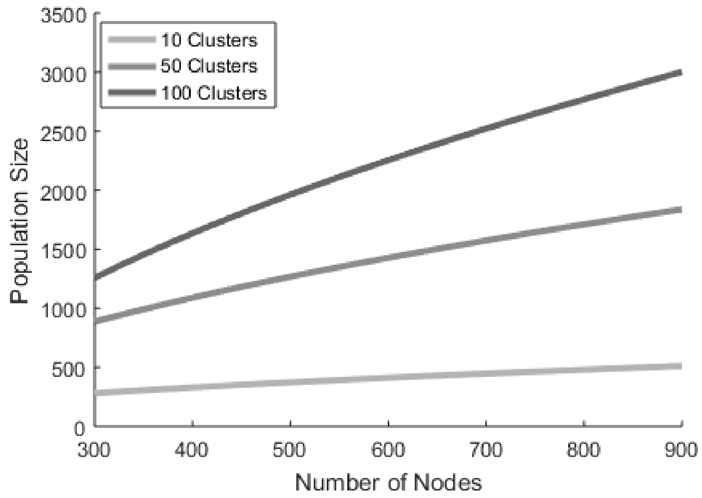

Figure 5.6: Population size with respect to problem instance size

In this algorithm, parent selection is completely random to promote diversity of the population. From two parents, one offspring is generated. In order not to miss diversity in early generations, number of offsprings have been set such that it is a small portion of the pool, which is determined by crossover probability α. For example, if $\alpha = 0.4$, number of offsprings is equal to 20% of the population size. Therefore, not all individuals have the chance to produce an offspring. Crossover logic is very similar to *Uniform Crossover*. Genes that both of the parents have are passed to the offspring. If a vertex exists twice in both parents (meaning that the vertex is the center of two clusters), it exists twice in the offspring as well. For each gene of the rest of the chromosome, a random number $\in [0, 1]$ is generated. If the number is less than 0.5, gene of parent 1 is passed to offspring. Otherwise, gene of parent 2 is passed to offspring. Pseudocode of the algorithm is given in Algorithm 5 for further details.

Local Search

In HGA-N, only vertices are searched. This Local Search operation has been designed to improve population (promote intensification, or exploitation) by improving fitter offsprings. The reason behind choosing fitter offsprings is to perform local search with better initial solutions. In most of the heuristics, initial solution quality affects both solution time and quality. Therefore, we select one best offspring in each

Algorithm 5 Crossover in HGA-N

```
Input: Population
Output: Offsprings
Shuffle the chromosomes in Population and store them in ShuffledParentList
NumberOfMatches = ⌊α * PopulationSize/2⌋, offsprings = ∅
for i = 1 to NumberOfMatches do
    Parent_1 = i^th chromosome of ShuffledParentList
    Parent_2 = i + 1^th chromosome of ShuffledParentList
    BothHave = Parent_1 ∩ Parent_2
    NewOffspring = BothHave
    Find number of genes to be added NumberToAdd
    Remove genes in BothHave from Parent_1 and Parent2
    for genes j=1 to NumberToAdd do
        Generate a random number r ∈ [0, 1]
        if r ≤ 0.5 then
            Add Parent_1(j) to NewOffspring
        else
            Add Parent_2(j) to NewOffspring
        end if
    end for
    Sort genes of NewOffspring in ascending order
    Add NewOffspring to Offsprings
    i = i + 2
end for
```

generation. The logic behind choosing one offspring is to minimize the number of local search operations to be performed. Moreover, with this way, we make sure that the population is improving without harming the diversity.

The procedure starts with the fittest offspring. First, genes of the offspring are shuffled since the cluster order is important in the improvement procedure. After shuffling, first cluster center is selected for improvement. Adjacent vertices (vertices that are connected to the current cluster center) are visited to find a better solution (a solution with a lower fitness function value). If an improved solution is found, the cluster center is updated, and unvisited adjacent vertices of the updated cluster center are visited. This procedure continues until the current cluster center does not improve. Then, the next cluster center is selected for improvement search. The procedure continues until the number of selected cluster centers exceeds $\lceil \beta * p \rceil$, which is discussed earlier in §5.2. The pseudocode is provided in Algorithm 6.

Generation Replacement

In the Generation Replacement step, population is created again from the pool of individuals that contain the current population and the offsprings. In a generation replacement stage, it is basically aimed to eliminate "not good" solutions which refers to the ones that do not have a potential to generate good offsprings. Therefore, it is desirable to remove weak solutions from the population. Meanwhile, we also want to eliminate similar individual groups (that is, we do not want a family to dominate the population), which have negative effects on diversity. The designed operation could be described as "Kill the weakest family member". If an offspring is weaker (has a higher objective function or lower fitness value than its parents), it does not enter the population. Otherwise, the weakest parent is replaced with the offspring. This operation is performed until each offspring is checked. Pseudocode of this step is given in Algorithm 7.

Algorithm 6 Local Search in HGA-N

Input: $Offspring, \beta$
Output: $ImprovedOffspring$
Shuffle genes of the $Offspring$
$ImprovedOffspring = Offspring$
Set $i = 1$
Set $Visited = \emptyset$
while do $i \leq \lceil \beta * p \rceil$
  Find $nAdj$, adjacent vertices to $Offspring_i$
  Find $nToVisit = nAdj \setminus Visited$
  if $nToVisit = \emptyset$ **then**
    $i = i + 1$
    Continue
  end if
  Find the best fitness value f_{best} from $nToVisit$ and n_{best}
  $Visited = Visited \cup nToVisit$
  if $f_{best} < f_{current} - \epsilon$ **then**
    $ImprovedOffspring_i = n_{best}$
    $f_{current} = f_{best}$
  else
    $i = i + 1$
  end if
end while
$ImprovedOffspring = Offspring$

Algorithm 7 Generation Replacement

Input:$Population, Offsprings, Parents$
Output:$NewPopulation$
for each offspring **do**
  if $f_{offspring} \leq \max\{f_{parent_1}, f_{parent_2}\}$ **then**
    Replace weaker parent with *offspring*
  end if
end for

Termination Condition

In this algorithm, a two-level termination condition is considered. The first condition is the number of generations. If it is less than $\sqrt{n}$, the number of vertices, the algorithm continues. Otherwise, population averages of the fitness values (denoted by $\bar{f}$) in the consecutive generations are compared. If $\bar{f}$ is changed less than $\delta\%$, the algorithm terminates. Here, δ is a parameter defined by the user. The first condition is to force the algorithm to iterate until sufficient number of new offsprings have been generated. If only the percentage improvement in $\bar{f}$ is checked, a premature convergence may occur. In the earlier iterations, because of the high variance in the population fitness function values, it is possible that there is no high improvement. Therefore, we need the algorithm to run for at least $\sqrt{n}$ generations. After that, by checking percentage improvement in $\bar{f}$, we try to measure population diversity. If the average does not change, population does not change. This implies that there is no room for further improvement, so the algorithm terminates.

5.3.2 Edge Based Hybrid Genetic Algorithm (HGA-E)

This approach differs from HGA-N in that the center locations could be anywhere on the graph $\mathbf{G}$. Therefore, this algorithm needs a more complex chromosome representation, and operators need to be adjusted accordingly. Especially, LS procedure is tailored accordingly. This algorithm has been designed especially for the problems that have optimal solutions along the edges, such as Fuzzy Clustering and Sum of Squares Clustering Problem.

It is worth noting that HGA-E uses the same Generation Replacement operation and Termination Condition as HGA-N. Therefore, these will not be explained again in this subsection to avoid repetition.

Chromosome Representation

In HGA-E, different from HGA-N, an edge based optimization is aimed. As in HGA-N, number of clusters p is a predefined value. The following representation scheme is tailored. As in HGA-N, with this scheme, the condition that the number of clusters is exactly p is guaranteed. Chromosome representation has been visualized in Figure 5.7 . The chromosome is composed of three arrays each containing p values representing vertex indices of an edge and position of each center. Two of them (referred as $vertex1$ and $vertex2$) store vertices that define the edge, and the third array (referred as $position$) stores position of center on the edge (normalized distance of the center from the first vertex). For example, if r_1 is equal to 0, x_1^1 is the location of cluster center 1. If r_1 is equal to 1, x_1^2 is the location of cluster center 1.

$vertex1$	x_1^1	x_2^1	...	x_p^1
$vertex2$	x_1^2	x_2^2	...	x_p^2
$position$	r_1	r_2	...	r_p

Figure 5.7: Chromosome Representation for HGA-E

Initial Population Generation

This procedure is very similar to that of HGA-N's. The main difference is that we generate random individuals by randomly generated edges instead of vertices. As in HGA-N, we guarantee that no duplicate solutions are generated until the population size exceeds $u_{max} = \binom{|E|}{p}/2$, where $|E|$ is number of edges and p is number of clusters. Initial position in the chromosome is not critical for the algorithm, since LS procedure is designed to find the best position. Therefore, position chromosome is set to 0.5 for all cluster centers. The population size used in the HGA-E depends on the size of problem instance with the formulation of (5.1) given in HGA-N. As in HGA-N, number of vertices is used to take the size of the problem instance into account. Since we work on edges, using number of edges instead of number of vertices in (5.1) could be another option. However, it is observed that the population size gets too large without a significant improvement in solution quality, but a significant increase in computation time. Therefore, in HGA-E, number of vertices is used to calculate the population size.

Crossover

Uniform crossover is performed in HGA-E similar to HGA-N. Parents are selected randomly, and one offspring is created from two parents. After selecting the parents, the edges that exist in both parents are found and passed to the offspring. Position values on these edges are randomly determined by $U[0,1]$, which refer to the convex combinations of positions in two parents. Then, a mapping procedure is called where indices of cluster centers are changed in both parents so that adjacent edges are in the same index in the chromosome and only one of them could be passed to the offspring. After mapping, as in HGA-N, a random number $\in [0,1]$ is generated for each of the rest of the genes. For each gene, being less than 0.5, the offspring takes gene of parent 1. Otherwise, it takes gene of parent 2. Pseudocode is given in Algorithm 8.

Local Search

Basic working principle of LS is as described in Section 5.2. Additional to this framework, searching the best solutions on the edge is performed in this operation. Therefore, in HGA-E, LS is essential to search the better solutions along the edges. Similar to HGA-N, after crossover operation is finished, the offspring that has the lowest fitness function value is selected. Then, LS procedure has been implemented. In this LS, as an additional feature, an array $ReferenceList$ is defined to store vertices. This list is needed to store end vertices, adjacent edges of whom will be searched. Adjacent edges of the vertices in the list will be visited as long as the cluster center is improved. This is necessary because even if we find an interior point on an edge as the best solution on that edge, we want to check other edges that are adjacent to end vertices of the current edge. Even if an edge does not contain a good solution, an end vertex contains a potentially good solution (a solution with fitness value equal to the best fitness value found) is added to $ReferenceList$.

Algorithm 9 starts with shuffling the cluster centers in $Offspring$. The loop starts with the first cluster center, and continues until $\lceil \beta * p \rceil$ cluster centers are visited for improvement. In the first iteration, the best location for the first cluster center is found with **Edge Search** procedure. If the best fitness value is worse than the current

Algorithm 8 Crossover in HGA-E

Input: $Population$
Output: $Offsprings$
Shuffle the chromosomes in $Population$ and store them in $ShuffledParentList$
$NumberOfMatches = \lfloor \alpha * PopulationSize/2 \rfloor$
$offsprings = \emptyset$
for $i = 1$ to $NumberOfMatches$ **do**
    $Parent_1 = i^{th}$ chromosome of $ShuffledParentList$
    $Parent_2 = i + 1^{th}$ chromosome of $ShuffledParentList$
    $BothHave = Parent_1 \cap Parent_2$
    Take convex combination of two positions in $Parent_1$ and $Parent_2$
    $NewOffspring = BothHave$
    Find number of genes to be added $NumberToAdd$
    Remove genes in $BothHave$ from $Parent_1$ and $Parent2$
    Map mutual edges of parents to same indices
    for genes j=1 to $NumberToAdd$ **do**
        Generate a random number $r \in [0, 1]$
        if $r \leq 0.5$ **then**
            Add $Parent_1(j)$ to $NewOffspring$
        else
            Add $Parent_2(j)$ to $NewOffspring$
        end if
    end for
    Sort genes of $NewOffspring$ in ascending order
    Add $NewOffspring$ to $Offsprings$
    $i = i + 2$
end for

fitness value, the center location is not updated and cluster center 2 is checked for improvement in the next iteration. Otherwise, location of the center 1 is updated and checked for improvement. For further improvement, the next iteration continues with cluster center 1. And the edges to be visited is calculated by using vertices in the $ReferenceList$. The pseudocode is given in Algorithm 9.

Edge Search

The key operation inside Local Search is *Edge Search*. Edge Search subroutine is designed to find the center location on an edge with minimum fitness value. The method to find this location differ with respect to problems and their properties. We have two main problems on hand to solve with HGA-E: Sum of Squares Clustering (SSC) and Fuzzy Clustering (FC).

In SSC problem, the objective function is a piecewise function at the arc bottleneck points and assignment bottleneck points. Given a solution, FDS is found by using these bottleneck points. First, subintervals are found by taking union of the arc and assignment bottleneck points on an edge and dividing the edge so that a subinterval does not contain any bottleneck points. In each subinterval, the fitness function is convex. Using the convexity structure, a local minimum point within each subinterval is found. Otherwise, one of the endpoints of the subinterval has the minimum fitness value. On an edge, for each subinterval, this calculation is made which forms the set of candidate solutions, and the best location on the edge is found.

In Fuzzy Clustering, similar to SSC, the objective function is a piecewise function at only the arc bottleneck points. First, subintervals on the edge is calculated by using arc bottleneck points. Then, in each subinterval, the minimum point is found. Since objective function of FC is nonconvex and a closed form formula for the derivative is harder to find than that of SSC, a derivative-free search method has been implemented. In this approach, we use Golden Section Search. After finding minimum points in each subinterval, the best location on the edge is found.

Algorithm 9 Local Search in HGA-E

Input: $Offspring, \beta$
Output: $ImprovedOffspring$
Shuffle genes of the $Offspring$
$ImprovedOffspring = Offspring$
Set $i = 1, Visited = \emptyset$
$f_{current} = Fitness(Offspring)$
Find $eAdj$, adjacent edges to $Offspring_1$
Find $eToVisit = eAdj - Visited$
while $i \leq \lceil \beta * p \rceil$ **do**
    if $eToVisit = \emptyset$ **then**
        $i = i + 1$
        Find $eAdj$, adjacent edges to vertices in $ImprovedOffspring_i$
        $Visited = \emptyset$
        $ReferenceList = \emptyset$
        Find $eAdj$, adjacent edges to $ImprovedOffspring_i$
        $eToVisit = eAdj$
        Continue
    end if

Algorithm 9 Local Search in HGA-E (continued)

Set $FDS = \emptyset$
for each edge in $eToVisit$ **do**
Set v_p=End Vertex 1
Set v_q=End Vertex 2
$FDS(v_p, v_q) =$ **EdgeSearch**(v_p, v_q)
$Visited = Visited \cup (v_p, v_q)$
$fit(p) = Fitness(v_p)$
$fit(q) = Fitness(v_q)$
end for
Find x_{best} and f_{best} from FDS
for each positive index i in fit **do**
if then$fit(v_i) \leq f_{best} + 10^{-5}$
Add v_i to $ReferenceList$
end if
end for
if $f_{best} < f_{current}$ **then**
Update offspring $ImprovedOffspring_i = x_{best}$
$f_{current} = f_{best}$
Find $eAdj$, adjacent edges to vertices in $ReferenceList$
$eToVisit = eAdj - Visited$
else
$i = i + 1$
$Visited = \emptyset$
$ReferenceList = \emptyset$
Find $eAdj$, adjacent edges to $ImprovedOffspring_i$
$eToVisit = eAdj$
end if
end while

CHAPTER 6

COMPUTATIONAL RESULTS

In order to analyze performance of HGA-N and HGA-E algorithms, computational studies are performed, outputs of which are discussed in this chapter. Four problems on hand, which are P-Median Problem, Sum of Squares Clustering Problem, PD-Clustering Problem and Fuzzy Clustering Problem are solved by using the proposed solution approaches. Three data sources have been used in the analysis. The first data set is from ORLib, problem instances for Uncapacitated P-Median Problem. ORLib instances contain 40 problems varying in size and network structure. The other two data sets are simulated using two different algorithms which will be described in §6.4. Each data set contains 60 problems varying in size and network structure. According to theoretical results derived in Chapter 4, either HGA-N or HGA-E is proposed as solution approach to each problem. Since it is shown that optimal solutions of P-Median Problem and PD-Clustering Problem is on vertices, HGA-N algorithm is proposed as solution approach. For P-Median Problem, optimal values of ORLib are also reported in the literature. Therefore, HGA-N results are compared with optimal values for P-Median Problem. PD-Clustering Problem is newly defined on networks; as a result, no reported solutions are available in the literature. Therefore, based on *PD-Clustering*, a heuristic which is named M-*PD-Clustering* is implemented to be able to do comparison. Since this heuristic needs vertex coordinates, two data sets which are mentioned have been simulated. For Sum of Squares Clustering Problem and Fuzzy Clustering Problem, based on the theoretical results, HGA-E is proposed. For Sum of Squares Clustering Problem with ORLib instances, optimal solutions are not known, but previously reported solutions by [1] is available. Therefore, for Sum of Squares Clustering Problem, these values have been used for comparison. Furthermore, to see the benefit of locating cluster centers on edges, HGA-N and HGA-E are

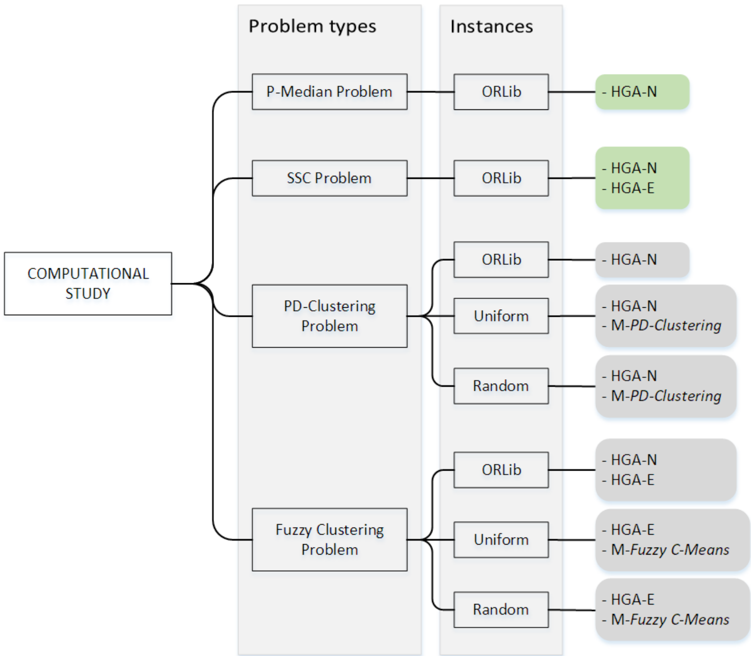

Figure 6.1: Representation of the computational studies made for each problem (green boxes indicate that there are solutions available in the literature)

also compared. Similar to PD-Clustering Problem, Fuzzy Clustering Problem also does not have reported solutions in the literature. Therefore, simulated data sets and a modified *Fuzzy C-Means* heuristic which is called M-*Fuzzy C-Means* (M-*FCM*) is used for comparison with HGA-E. As in Sum of Squares Clustering Problem, to see the benefit of locating centers on edges, HGA-N and HGA-E comparison is also made. Computational studies could be summarized as in Figure 6.1.

This chapter is organized as follows. First, parameter settings and computation environment is provided in §6.1. Then, analysis for hard assignment problems will be provided in §6.2. After discussing computational results for soft assignment problems in §6.3 and §6.4, an interesting property discovered for soft assignment problems will be discussed in §6.5.

6.1 Parameter Settings and Environment

Parameter settings for both HGA-N and HGA-E are given in Table 6.1 along with their descriptions. For all problem instances and all four types of problems solved, same settings were used. These settings are determined with preliminary runs performed on selected problem instances. All algorithms are coded in MATLAB R2017a. Computational studies are conducted on a PC with a 3.6 GHz Intel Core i7-4790HQ processor and 8 GB of RAM.

Table 6.1: Parameter settings for HGA-N and HGA-E

Parameter	Description	Value
α	Crossover Probability	0.4
β	Portion determining the number of cluster centers that undergo LS procedure $\in [0, 1]$	0.7
δ	Stopping condition for percentage improvement between mean fitness values of consecutive generations $\in [0, 100]$	10^{-5}

6.2 Hard Assignment Problems

In this section, performance of HGA-N and HGA-E in hard assignment problems will be analyzed. First, solutions obtained with our algorithm will be compared with the solutions reported in the literature. Then, best solutions of both P-Median Problem and Sum of Squares Clustering Problem will be analyzed using selected problem instances.

6.2.1 Comparison with Literature using ORLib Instances

As discussed before, in the scope of this study, we considered two hard assignment problems, P-Median Problem and Sum of Squares Clustering (SSC) Problem. In this subsection, solutions obtained for these problems with ORLib instances will be reported and compared with the solutions reported in the literature.

P-Median Problem

For P-Median Problem, it was proven that the optimal cluster center locations will always be on vertices. Therefore, only HGA-N algorithm is executed for this problem. Results are given in Table 6.2 along with the instance size (number of vertices, clusters and edges of the network), optimal objective function values found by the algorithm, and best, average and worst percentage deviations of HGA-N from the optimal objective function values reported in the literature. For each problem instance, 5 replications were made. Reported computational times are total times of these 5 replications. 0.00% deviations are shown as "-" in the tables.

Checking the results in the Table 6.2, runtime increases as instance size increases. It increases especially with the number of clusters. In 39 instances out of 40, HGA-N was able to find the optimal solution. In the largest instance which is ORLib 40, the algorithm deviates from the best solution by 0.04 %, which could be considered as insignificant. Average of the average and worst percentage deviations are 0.02 % and 0.04 %, respectively, which shows that the algorithm is stable. In other words, the variance within replications is low. In short, we can say that HGA-N performs well in ORLib problem instances when P-Median Problem is considered.

SSC Problem

Unlike P-Median Problem, SSC Problem can have cluster center locations on not only the vertices, but also on the edges. Therefore, HGA-E algorithm is applied to solve SSC Problem. Additionally, to see the amount of improvement in objective function value, HGA-N is also executed, and HGA-N and HGA-E solutions are compared. As stated previously, optimal solutions are not known for Sum of Squares Clustering Problem. Therefore, the solutions obtained with HGA-N and HGA-E are compared with the solutions reported by [1]. Results found with HGA-N and HGA-E are given in Tables 6.3 and 6.4, respectively. In these tables, instance size (number of vertices, clusters and edges), optimal objective function values, and best, average and worst percentage deviations of the algorithm from the objective function values reported in [1] are reported. As in P-Median Problem, 5 replications were made for each problem

Table 6.2: Results of HGA-N for P-Median Problem and comparison with optimal solutions

Instance	Vertices-Clusters-Edges	Optimal Value	Best % Dev	Avg % Dev	Worst % Dev	Runtime (sec)
ORLib1	100-5-198	5819.00	-	-	-	0.52
ORLib2	100-10-193	4093.00	-	-	-	1.53
ORLib3	100-10-198	4250.00	-	-	-	1.31
ORLib4	100-20-196	3034.00	-	-	-	4.86
ORLib5	100-33-196	1355.00	-	0.07	0.22	10.60
ORLib6	200-5-786	7824.00	-	-	-	0.79
ORLib7	200-10-779	5631.00	-	-	-	2.59
ORLib8	200-20-792	4445.00	-	-	-	8.40
ORLib9	200-40-785	2734.00	-	0.18	0.48	35.38
ORLib10	200-67-786	1255.00	-	0.08	0.32	82.74
ORLib11	300-5-1772	7696.00	-	-	-	1.13
ORLib12	300-10-1758	6634.00	-	-	-	3.59
ORLib13	300-30-1760	4374.00	-	-	-	25.22
ORLib14	300-60-1771	2968.00	-	-	0.03	99.54
ORLib15	300-100-1754	1729.00	-	0.06	0.06	366.52
ORLib16	400-5-3153	8162.00	-	-	-	1.59
ORLib17	400-10-3142	6999.00	-	-	-	4.95
ORLib18	400-40-3134	4809.00	-	0.02	0.04	81.69
ORLib19	400-80-3134	2845.00	-	0.04	0.07	370.61
ORLib20	400-133-3144	1789.00	-	-	-	939.49
ORLib21	500-5-4909	9138.00	-	-	-	1.69
ORLib22	500-10-4896	8579.00	-	-	-	6.76
ORLib23	500-50-4903	4619.00	-	-	-	180.00
ORLib24	500-100-4914	2961.00	-	-	-	662.24
ORLib25	500-167-4894	1828.00	-	-	-	2288.24
ORLib26	600-5-7069	9917.00	-	0.01	0.07	2.90
ORLib27	600-10-7072	8307.00	-	-	-	10.65
ORLib28	600-60-7054	4498.00	-	-	0.02	332.01
ORLib29	600-120-7042	3033.00	-	0.03	0.03	1242.94
ORLib30	600-200-7042	1989.00	-	0.05	0.20	5364.87
ORLib31	700-5-9601	10086.00	-	-	-	3.07
ORLib32	700-10-9584	9297.00	-	-	-	11.08
ORLib33	700-70-9616	4700.00	-	-	0.02	577.19
ORLib34	700-140-9585	3013.00	-	-	-	2148.77
ORLib35	800-5-12548	10400.00	-	-	-	3.76
ORLib36	800-10-12560	9934.00	-	-	-	15.44
ORLib37	800-80-12564	5057.00	-	0.02	0.04	844.96
ORLib38	900-5-15898	11060.00	-	-	-	6.49
ORLib39	900-10-15896	9423.00	-	-	-	15.68
ORLib40	900-90-15879	5128.00	0.04	0.06	0.08	1360.26
	Average		**0.00**	**0.02**	**0.04**	**428.05**

instance. In the Tables 6.3 and 6.4, negative deviations are shown in boldface showing that HGA obtains better solutions than the reported ones.

When the HGA-N results in Table 6.3 is analyzed, it could be seen that the runtime increases as number of vertices and number of clusters increases. In 4 instances, HGA-N finds better solution than the ones in [1]. In ORLib 6, the highest percentage deviation is observed which is 0.47 %. In 8 problem instances, HGA-N found solutions with the objective function values higher than the reported values. On average, HGA-N deviates 0.03%, which is considerably low. An interesting observation is that as the number of clusters increases, HGA-N finds better solutions than those reported in [1], which could be a sign of that the algorithm in [1] gets stuck at local solutions as number of clusters increases.

Regarding HGA-E results in Table 6.4, as in HGA-N, it is observed that the runtime increases as the size of the problem instance increases. As a matter of fact, the increase in runtime as the number of clusters increase is more than that in HGA-N. A possible reason could be that edge search operation is computationally expensive. Regarding the solution quality, in 11 problem instances, HGA-E finds solutions with objective function value lower than the values reported in [1]. In 18 instances, HGA-E finds solutions that are worse than the reported solutions in terms of the objective function value. Overall, in 22 instances, HGA-E finds solutions at least as good as the solutions in [1]. Average best deviation over 40 instances is 0.45 %, average is 0.82 % and worst is 1.29 %. In one instance which is ORLib 5, the highest percentage deviation in the best deviation is observed, which is 5.06 %. However, without loss of generality, it could be said that HGA-E has a promising performance in ORLib problem instances when SSC Problem is solved.

Solution Analysis of HGA-N and HGA-E in SSC

In Chapter 4, it has been shown that centers can be located on the edges in the optimal solution of SSC Problem. Still, in order to observe the loss in the objective function value when centers are only allowed to be located on vertices, HGA-N is also executed. In the Table 6.5, the best objective function values found with HGA-E and HGA-N with 5 replications is reported. Additionally, percentage deviation of HGA-N

Table 6.3: Results of HGA-N for SSC Problem and comparison with the reported results in [1]

Instance	Vertices-Clusters-Edges	Best Known Value	Best % Dev	Avg % Dev	Worst % Dev	Runtime (sec)
ORLib1	100-5-198	450233.00	-	-	-	0.47
ORLib2	100-10-193	256874.00	-	-	-	1.89
ORLib3	100-10-198	263385.00	-	-	-	1.41
ORLib4	100-20-196	153963.00	-	0.14	0.32	4.95
ORLib5	100-33-196	42671.00	0.47	0.67	1.04	10.71
ORLib6	200-5-786	406195.00	-	-	-	0.79
ORLib7	200-10-779	221631.00	-	0.05	0.24	2.64
ORLib8	200-20-792	151558.00	-	0.08	0.39	8.74
ORLib9	200-40-785	66525.00	-	-	-	37.85
ORLib10	200-67-786	15938.00	-	-	-	111.26
ORLib11	300-5-1772	256532.00	-	-	-	1.04
ORLib12	300-10-1758	197814.00	-	-	0.01	3.88
ORLib13	300-30-1760	99210.00	-	0.06	0.13	32.65
ORLib14	300-60-1771	49977.00	-	0.33	0.76	141.25
ORLib15	300-100-1754	20213.00	-	-	-	409.63
ORLib16	400-5-3153	209886.00	-	-	-	1.36
ORLib17	400-10-3142	160401.00	-	-	-	5.08
ORLib18	400-40-3134	88234.00	-	0.19	0.43	131.96
ORLib19	400-80-3134	33782.00	-	0.07	0.37	449.14
ORLib20	400-133-3144	16032.00	**-0.06**	0.08	0.24	941.47
ORLib21	500-5-4909	203552.00	-	-	-	1.68
ORLib22	500-10-4896	188857.00	-	0.05	0.12	7.22
ORLib23	500-50-4903	66257.00	0.03	0.11	0.18	236.14
ORLib24	500-100-4914	29478.00	-	-	-	770.95
ORLib25	500-167-4894	13377.00	0.02	0.04	0.10	2523.86
ORLib26	600-5-7069	199503.00	-	-	-	2.74
ORLib27	600-10-7072	147096.00	-	-	-	10.21
ORLib28	600-60-7054	51239.00	0.05	0.09	0.13	303.33
ORLib29	600-120-7042	25848.00	0.07	0.14	0.22	1322.75
ORLib30	600-200-7042	12533.00	0.18	0.33	0.46	5830.65
ORLib31	700-5-9601	171963.00	-	-	-	3.53
ORLib32	700-10-9584	157177.00	-	0.01	0.07	10.78
ORLib33	700-70-9616	47255.00	**-0.04**	0.09	0.16	1086.41
ORLib34	700-140-9585	21981.00	**-0.09**	0.01	0.19	2806.83
ORLib35	800-5-12548	160564.00	-	-	-	3.74
ORLib36	800-10-12560	152914.00	-	0.02	0.08	15.64
ORLib37	800-80-12564	48246.00	0.24	0.28	0.35	1345.28
ORLib38	900-5-15898	161102.00	-	0.01	0.04	5.42
ORLib39	900-10-15896	125175.00	0.17	0.71	1.10	17.71
ORLib40	900-90-15879	43035.00	**-0.02**	0.03	0.19	1579.10
	Average		**0.03**	**0.09**	**0.18**	**504.55**

Table 6.4: Results of HGA-E for SSC Problem and comparison with the reported results in [1]

Instance	Vertices-Clusters-Edges	Best Known Value	Best % Dev	Avg % Dev	Worst % Dev	Runtime (sec)
ORLib1	100-5-198	450043.94	-	-	-	6.50
ORLib2	100-10-193	253067.60	0.60	0.73	0.93	25.52
ORLib3	100-10-198	259643.17	-	0.19	0.95	21.98
ORLib4	100-20-196	147685.50	**-0.34**	0.01	0.67	83.19
ORLib5	100-33-196	40066.36	**-2.22**	**-1.27**	1.51	156.40
ORLib6	200-5-786	386642.24	5.06	5.06	5.06	16.52
ORLib7	200-10-779	221602.83	**-0.13**	**-0.03**	-	65.81
ORLib8	200-20-792	151094.71	**-0.27**	**-0.24**	**-0.09**	262.36
ORLib9	200-40-785	63126.34	**-1.57**	**-1.48**	**-1.10**	937.98
ORLib10	200-67-786	14917.01	**-0.85**	0.41	2.34	1302.60
ORLib11	300-5-1772	256512.74	-	-	0.01	31.76
ORLib12	300-10-1758	197814.00	**-0.01**	**-0.01**	**-0.01**	104.55
ORLib13	300-30-1760	98471.40	**-0.06**	0.07	0.14	907.93
ORLib14	300-60-1771	49152.57	0.65	0.99	1.76	2671.20
ORLib15	300-100-1754	18653.68	**-0.08**	0.40	1.03	5896.32
ORLib16	400-5-3153	209886.00	-	0.08	0.27	40.28
ORLib17	400-10-3142	160401.00	-	0.11	0.55	127.40
ORLib18	400-40-3134	87499.01	**-0.14**	1.15	1.84	2578.46
ORLib19	400-80-3134	32292.46	0.84	1.42	1.78	7089.94
ORLib20	400-133-3144	14930.55	1.63	2.63	3.58	16030.88
ORLib21	500-5-4909	203552.00	-	-	-	57.77
ORLib22	500-10-4896	188857.00	-	0.08	0.12	206.61
ORLib23	500-50-4903	65834.72	0.76	0.99	1.29	4845.14
ORLib24	500-100-4914	28533.80	0.79	1.30	1.66	16319.26
ORLib25	500-167-4894	12502.12	**-0.52**	1.43	2.39	33039.35
ORLib26	600-5-7069	199503.00	-	0.19	0.93	76.22
ORLib27	600-10-7072	147096.00	-	-	-	240.18
ORLib28	600-60-7054	51030.45	0.95	1.40	1.92	8349.54
ORLib29	600-120-7042	25335.36	1.56	2.09	3.19	26971.91
ORLib30	600-200-7042	11671.78	3.85	4.67	5.42	61492.92
ORLib31	700-5-9601	171963.00	-	0.23	0.57	113.06
ORLib32	700-10-9584	157177.00	-	0.04	0.07	309.84
ORLib33	700-70-9616	47188.77	0.96	1.25	1.62	15883.95
ORLib34	700-140-9585	21461.21	2.77	3.41	3.85	47305.57
ORLib35	800-5-12548	160541.91	-	0.22	1.05	133.58
ORLib36	800-10-12560	152914.00	-	0.03	0.08	390.50
ORLib37	800-80-12564	48195.16	1.74	1.99	2.36	26577.95
ORLib38	900-5-15898	161102.00	-	0.05	0.11	195.14
ORLib39	900-10-15896	125175.00	0.17	0.91	1.10	464.11
ORLib40	900-90-15879	42877.82	1.93	2.30	2.64	37005.62
	Average		**0.45**	**0.82**	**1.29**	**7958.39**

from HGA-E is reported (the last column in Table 6.5). When this table is analysed in detail, it could be seen that as number of clusters increases, locating centers to edges makes more difference in the objective function value. Additionally, as instance size increases, locating centers to Vertices does not cause large deviations in objective function value. It could be noticed that in some of the instances, locating centers to Vertices leads to a negative deviation, that is, a better solution. This is probably due to the fact that HGA-E obtained a local optimal solution. In 12 instances, HGA-N and HGA-E obtained the solutions with the same objective function value. The average percentage deviation over 40 instances are 1.55.

6.2.2 Comparison of Center Locations

When the same problem instance is solved with two hard clustering problems which have different objective functions, center locations may change. In order to see this difference, center locations of ORLib instances with 5 and 10 clusters that are obtained with HGA-N is analyzed. The center locations for both P-Median Problem and SSC Problem are obtained with HGA-N algorithm. The results are presented in Table 6.6.

Among 9 instances with 5 clusters, 6 of them have common cluster centers regardless of the problem type. Out of 10 instances with 10 clusters, only one instance has common cluster centers in solutions obtained for both problems. On average, in the instances with 5 clusters, 4 cluster centers are common, and in the instances with 10 clusters, 7 cluster centers are common.

Observed difference between P-Median Problem and SSC Problem highly depends on characteristics of instances. Since SSC uses squared distance, it penalizes vertices that could be considered as outliers more than P-Median Problem. Also, it is observed that as number of clusters increases, portion of common centers decreases. This is expected because as the number of clusters increases and clusters get smaller, each cluster center could be affected by outlier vertices more. Therefore, the observed results seems reasonable and justifiable.

Table 6.5: Best objective function values found with HGA-E and HGA-N and percentage deviation of HGA-N from HGA-E

Instance	Vertices-Clusters-Edges	HGA-E	HGA-N	% Dev
ORLib1	100-5-198	450043.94	450233.00	0.04
ORLib2	100-10-193	254579.18	256874.00	0.90
ORLib3	100-10-198	259643.17	263385.00	1.44
ORLib4	100-20-196	147186.47	153963.00	4.60
ORLib5	100-33-196	39175.52	42870.00	9.43
ORLib6	200-5-786	406195	406195.00	-
ORLib7	200-10-779	221309.27	221631.00	0.15
ORLib8	200-20-792	150680.44	151558.00	0.58
ORLib9	200-40-785	62135.79	66525.00	7.06
ORLib10	200-67-786	14790.82	15938.00	7.76
ORLib11	300-5-1772	256512.74	256532.00	0.01
ORLib12	300-10-1758	197791.68	197814.00	0.01
ORLib13	300-30-1760	98414.21	99210.00	0.81
ORLib14	300-60-1771	49471.62	49977.00	1.02
ORLib15	300-100-1754	18639.4	20213.00	8.44
ORLib16	400-5-3153	209886	209886.00	-
ORLib17	400-10-3142	160401	160401.00	-
ORLib18	400-40-3134	87381.12	88234.00	0.98
ORLib19	400-80-3134	32562.53	33782.00	3.75
ORLib20	400-133-3144	15174.26	16023.00	5.59
ORLib21	500-5-4909	203552	203552.00	-
ORLib22	500-10-4896	188857	188857.00	-
ORLib23	500-50-4903	66335.9	66276.00	-0.09
ORLib24	500-100-4914	28759.3	29478.00	2.50
ORLib25	500-167-4894	12436.95	13380.00	7.58
ORLib26	600-5-7069	199503	199503.00	-
ORLib27	600-10-7072	147096	147096.00	-
ORLib28	600-60-7054	51515.92	51265.00	-0.49
ORLib29	600-120-7042	25731.59	25867.00	0.53
ORLib30	600-200-7042	12121.27	12556.00	3.59
ORLib31	700-5-9601	171963	171963.00	-
ORLib32	700-10-9584	157173.3	157177.00	-
ORLib33	700-70-9616	47642.67	47236.00	-0.85
ORLib34	700-140-9585	22055.7	21961.00	-0.43
ORLib35	800-5-12548	160543.07	160564.00	0.01
ORLib36	800-10-12560	152911.84	152914.00	-
ORLib37	800-80-12564	49034.58	48361.00	-1.37
ORLib38	900-5-15898	161102	161102.00	-
ORLib39	900-10-15896	125382	125382.00	-
ORLib40	900-90-15879	43703.95	43026.00	-1.55
	Average			1.55

Table 6.6: Comparison of center locations of P-Median Problem and SSC Problem

Instance	Vertices-Clusters-Edges	Number of Common Centers
ORLib1	100-5-198	5
ORLib2	100-10-193	4
ORLib3	100-10-198	7
ORLib6	200-5-786	2
ORLib7	200-10-779	7
ORLib11	300-5-1772	5
ORLib12	300-10-1758	7
ORLib16	400-5-3153	1
ORLib17	400-10-3142	10
ORLib21	500-5-4909	5
ORLib22	500-10-4896	7
ORLib26	600-5-7069	5
ORLib27	600-10-7072	7
ORLib31	700-5-9601	3
ORLib32	700-10-9584	5
ORLib35	800-5-12548	5
ORLib36	800-10-12560	7
ORLib38	900-5-15898	5
ORLib39	900-10-15896	9
Average		5.58
	5 Clusters	4.00
	10 Clusters	7.00

6.3 Soft Assignment Problems

In this section, analysis and interpretation of results of HGA-N and HGA-E regarding soft clustering problems, namely PD-Clustering and Fuzzy Clustering Problem, will be focused.

6.3.1 Solutions with ORLib Instances

In this subsection, solutions obtained by HGA-N and HGA-E algorithms for ORLib problem instances will be reported and discussed. Since there are no previously reported solutions in the literature, performance will be evaluated internally.

PD-Clustering Problem

For PD-Clustering Problem, it has been proven in Chapter 4 that the optimal cluster centers will always be on vertices. Therefore, HGA-N algorithm is executed for this problem. For each problem instance, 5 replications were made. The results are presented in Table 6.7 with best found objective function value, average and worst percentage deviations from the best found, and runtime values, which is the summation of runtimes of 5 replications. To begin with, runtimes for PD-Clustering Problem is relatively low. This is because of that there are many alternative solutions observed on the network which decreases number of generations needed. In 21 problem instances out of 40, the same objective function value has been obtained in all replications. For the remaining, percentage deviations are relatively low. The average of average and worst deviations over 40 instances are 0.00% and 0.02 %, respectively. This shows us that HGA-N has a stable performance on ORLib instances when PD-Clustering Problem is solved.

Fuzzy Clustering Problem

For Fuzzy Clustering Problem on networks, in Chapter 4, it has been discussed that cluster centers could be located on the edges at optimal solutions. Therefore, HGA-E

Table 6.7: Results of HGA-N for PD-Clustering Problem

Instance	Vertices-Clusters-Edges	Best Found Value	Avg % Dev	Worst % Dev	Runtime (sec)
ORLib1	100-5-198	1841.95	-	-	0.40
ORLib2	100-10-193	844.32	-	-	0.82
ORLib3	100-10-198	906.11	-	-	0.82
ORLib4	100-20-196	456.00	-	-	2.42
ORLib5	100-33-196	180.01	0.01	0.03	8.80
ORLib6	200-5-786	2219.57	-	-	0.51
ORLib7	200-10-779	985.39	-	-	1.38
ORLib8	200-20-792	536.14	-	-	5.28
ORLib9	200-40-785	235.82	0.01	0.01	16.58
ORLib10	200-67-786	98.93	-	-	46.53
ORLib11	300-5-1772	2022.65	-	-	0.68
ORLib12	300-10-1758	1160.93	-	-	2.17
ORLib13	300-30-1760	353.09	0.01	0.02	18.92
ORLib14	300-60-1771	180.80	-	-	58.79
ORLib15	300-100-1754	88.35	0.01	0.01	166.43
ORLib16	400-5-3153	2222.81	-	-	1.03
ORLib17	400-10-3142	1175.45	0.01	0.02	3.86
ORLib18	400-40-3134	317.64	-	-	41.20
ORLib19	400-80-3134	141.01	-	-	141.38
ORLib20	400-133-3144	78.38	-	0.01	467.18
ORLib21	500-5-4909	2502.80	0.05	0.24	1.23
ORLib22	500-10-4896	1381.59	-	-	4.25
ORLib23	500-50-4903	259.73	0.01	0.01	90.30
ORLib24	500-100-4914	122.29	-	0.01	327.89
ORLib25	500-167-4894	62.89	-	0.01	860.06
ORLib26	600-5-7069	2625.66	-	-	1.49
ORLib27	600-10-7072	1266.08	-	-	5.21
ORLib28	600-60-7054	208.95	0.01	0.01	188.53
ORLib29	600-120-7042	104.25	-	-	594.85
ORLib30	600-200-7042	59.65	-	0.01	1690.93
ORLib31	700-5-9601	2754.00	0.02	0.12	2.14
ORLib32	700-10-9584	1471.59	-	-	6.35
ORLib33	700-70-9616	203.55	0.01	0.01	275.65
ORLib34	700-140-9585	93.44	-	-	1085.95
ORLib35	800-5-12548	2764.37	-	-	2.42
ORLib36	800-10-12560	1561.50	-	-	8.16
ORLib37	800-80-12564	199.17	-	0.01	502.79
ORLib38	900-5-15898	2944.85	0.02	0.12	2.88
ORLib39	900-10-15896	1448.67	-	-	8.82
ORLib40	900-90-15879	179.88	0.01	0.02	736.55
	Average		**0.00**	**0.02**	**184.54**

algorithm is executed for this problem. For the sake of comparison, HGA-N algorithm is also executed. First, performance of HGA-N and HGA-E algorithms will be discussed. Then, to see the difference between locating cluster centers to edges and vertices, HGA-E and HGA-N comparison will be discussed.

It could be said that HGA-N has relatively low runtimes for ORLib problem instances and Fuzzy Clustering Problem. In the Table 6.8, it could be observed that in 24 problem instances out of 40, same objective function value is obtained in all replications. In other 16 instances, percentage deviations are considerably low. Percentage deviations are observed as number of clusters and instance size increase.

When HGA-E solutions are considered, high computing times are observed, especially as the number of clusters increase. Results are presented in Table 6.9 along with the objective function values found with HGA-N. In eight instances, HGA-E found solutions with the same objective function value in 5 replications. In the remaining instances, percentage deviation increases as the number of clusters increases with the number of vertices. An interesting result is that in three instances, HGA-N obtained objective function values better than HGA-E with 0.01% deviation. These instances have something in common – all have the highest number of clusters among the instances with the same number of vertices. This interesting result could be a sign of that HGA-E may have difficuly in finding good solutions as size of solution space increases. These small deviations could also be a result of computation error. However, regarding drastic differences in runtimes, solving Fuzzy Clustering Problem for ORLib instances with HGA-N seems more reasonable.

6.4 Comparison of HGA with the Soft Clustering Heuristics

In this section, for PD-Clustering Problem and Fuzzy Clustering Problem, HGA approaches will be compared with heuristic approaches from the literature; namely *Fuzzy C-Means* and *PD-Clustering*. These heuristics solve corresponding problems on plane. Therefore, we modified these heuristics so that the solution is on the network. M-*PD-Clustering*, which is modified version of *PD-Clustering*, maps the solution obtained by *PD-Clustering* to vertices, where M-*Fuzzy C-Means*, modified

Table 6.8: Results of HGA-N for FC problem with m=3

Instance -	Vertices-Clusters-Edges	Best Found Value	Avg % Dev	Worst % Dev	Runtime (sec)
ORLib1	100-5-198	40814.86	-	-	0.42
ORLib2	100-10-193	9110.85	-	-	0.95
ORLib3	100-10-198	10549.46	-	-	0.84
ORLib4	100-20-196	2841.03	-	-	2.57
ORLib5	100-33-196	542.78	-	-	7.92
ORLib6	200-5-786	29140.97	-	-	0.50
ORLib7	200-10-779	5891.58	-	-	1.52
ORLib8	200-20-792	1775.70	0.01	0.02	5.71
ORLib9	200-40-785	386.81	-	-	18.07
ORLib10	200-67-786	78.45	0.01	0.02	39.94
ORLib11	300-5-1772	15919.02	-	-	0.73
ORLib12	300-10-1758	5129.39	-	-	2.23
ORLib13	300-30-1760	496.98	0.06	0.08	17.30
ORLib14	300-60-1771	149.06	0.02	0.03	63.48
ORLib15	300-100-1754	39.15	-	-	186.83
ORLib16	400-5-3153	14093.02	-	-	0.99
ORLib17	400-10-3142	3885.84	-	-	3.68
ORLib18	400-40-3134	296.71	0.01	0.02	50.71
ORLib19	400-80-3134	65.31	-	-	154.33
ORLib20	400-133-3144	24.00	-	-	463.40
ORLib21	500-5-4909	13921.86	-	-	1.32
ORLib22	500-10-4896	4258.49	-	-	4.20
ORLib23	500-50-4903	156.78	0.03	0.05	102.46
ORLib24	500-100-4914	38.01	-	0.01	368.66
ORLib25	500-167-4894	11.24	0.03	0.04	1227.23
ORLib26	600-5-7069	12936.60	-	-	1.52
ORLib27	600-10-7072	3043.99	-	-	5.22
ORLib28	600-60-7054	83.48	0.01	0.02	219.48
ORLib29	600-120-7042	22.70	-	-	769.72
ORLib30	600-200-7042	9.09	0.01	0.02	2139.90
ORLib31	700-5-9601	11836.62	0.07	0.35	2.21
ORLib32	700-10-9584	3454.42	-	0.01	6.58
ORLib33	700-70-9616	66.63	0.01	0.02	331.09
ORLib34	700-140-9585	15.57	0.01	0.01	1240.41
ORLib35	800-5-12548	10487.14	-	-	2.30
ORLib36	800-10-12560	3339.31	-	-	8.51
ORLib37	800-80-12564	55.87	-	0.01	533.32
ORLib38	900-5-15898	10648.72	-	-	3.01
ORLib39	900-10-15896	2598.34	-	-	9.27
ORLib40	900-90-15879	40.30	0.01	0.02	836.37
	Average		**0.01**	**0.02**	**0.02**

Table 6.9: Results of HGA-E for FC problem with m=3

Instance	Vertices-Clusters-Edges	Value (HGA-E)	Avg % Dev	Worst % Dev	HGA-N Dev	Runtime (sec)
ORLib1	100-5-198	40814.86	0.24	0.68	-	13.98
ORLib2	100-10-193	9110.83	-	-	-	36.83
ORLib3	100-10-198	10549.46	-	0.01	-	34.80
ORLib4	100-20-196	2841.23	0.01	0.01	-	167.52
ORLib5	100-33-196	542.11	0.27	0.78	-	412.33
ORLib6	200-5-786	29140.97	-	-	-	42.78
ORLib7	200-10-779	5891.58	-	0.01	-	96.01
ORLib8	200-20-792	1775.83	0.02	0.07	-	548.24
ORLib9	200-40-785	386.84	0.02	0.04	-	2061.17
ORLib10	200-67-786	78.61	0.17	0.35	-	8012.85
ORLib11	300-5-1772	15919.02	-	-	-	70.29
ORLib12	300-10-1758	5129.39	-	-	-	247.05
ORLib13	300-30-1760	496.90	0.01	0.01	-	1666.82
ORLib14	300-60-1771	149.28	0.03	0.06	-	11817.41
ORLib15	300-100-1754	39.23	0.07	0.16	-	25602.77
ORLib16	400-5-3153	14093.02	-	-	-	97.62
ORLib17	400-10-3142	3885.85	-	0.01	-	406.79
ORLib18	400-40-3134	296.99	0.02	0.04	-	6850.09
ORLib19	400-80-3134	65.46	0.03	0.05	-	24470.98
ORLib20	400-133-3144	24.28	0.16	0.26	-0.01	95963.86
ORLib21	500-5-4909	13921.86	0.11	0.57	-	142.71
ORLib22	500-10-4896	4258.49	0.03	0.17	-	499.77
ORLib23	500-50-4903	156.77	0.02	0.03	-	9838.71
ORLib24	500-100-4914	38.08	0.03	0.06	-	50642.39
ORLib25	500-167-4894	11.31	0.26	0.53	-0.01	189599.91
ORLib26	600-5-7069	12936.60	-	-	-	196.64
ORLib27	600-10-7072	3043.99	-	-	-	657.80
ORLib28	600-60-7054	83.46	0.02	0.06	-	19519.96
ORLib29	600-120-7042	22.71	0.04	0.08	-	98824.86
ORLib30	600-200-7042	9.16	0.22	0.36	-0.01	400567.11
ORLib31	700-5-9601	11836.62	-	-	-	202.88
ORLib32	700-10-9584	3454.78	0.02	0.04	-	816.55
ORLib33	700-70-9616	66.70	0.01	0.03	-	32608.48
ORLib34	700-140-9585	15.58	0.13	0.49	-	134356.54
ORLib35	800-5-12548	10487.14	0.05	0.18	-	308.85
ORLib36	800-10-12560	3339.53	0.01	0.02	-	1062.60
ORLib37	800-80-12564	55.92	0.05	0.11	-	60750.38
ORLib38	900-5-15898	10653.85	0.07	0.14	-	356.56
ORLib39	900-10-15896	2598.53	-	0.01	-	889.59
ORLib40	900-90-15879	40.31	0.03	0.05	-	80025.64
	Average		**0.05**	**0.14**	**0.00**	**31512.20**

version of *Fuzzy C-Means*, maps the solution of *Fuzzy C-Means* to the closest point on the network. To use these modified heuristics, coordinates of the vertices on plane are needed. Since ORLib instances do not have vertex coordinates, those instances cannot be used in these experiments. Therefore, new data sets have been simulated. Two data sets, each has 60 instances, have been simulated by using two different procedures.

In this section, data generation algorithms will be discussed first. Then, performance of HGA-N will be compared with PD-Clustering heuristic and performance of HGA-E will be compared with *Fuzzy C-Means* heuristic. Lastly, HGA-N and HGA-E solutions for simulated data sets in Fuzzy Clustering Problem will be discussed.

6.4.1 Data Generation for Heuristics

In Data Generation, two different data sets have been generated, namely *Uniform* and *Random*. In Figure 6.2, their structural difference could be observed visually. In *Uniform* instances, vertices look like equidistant to each other, while in *Random* instances, they are completely random. Connections are also more restricted in *Uniform* instances in a way that a vertex is mostly connected to its neighbors only. In the *Random* instances, the connections are random and there are fewer restrictions.

Uniform data set contains 60 problem instances with different number of vertices and edges. The generation procedure starts with generating random normal coordinates for vertices. Then, to ensure connectivity of the network, a random spanning tree is generated which also considers vertex proximity. After that, to satisfy the number of edges requirement, random edges are added to the network according to an insertion algorithm. Pseudocode is given in Appendix B.1.

Similarly, *Random* data set contains 60 problem instances with different number of vertices and edges. First, vertex coordinates are generated randomly. Second, to ensure connectivity, under the assumption that we have a complete graph, Prim's Algorithm is implemented to find a Minimum Spanning Tree [32]. Lastly, an edge insertion procedure is executed to ensure that we have as many number of edges as needed. Pseudocode is given in Appendix B.2.

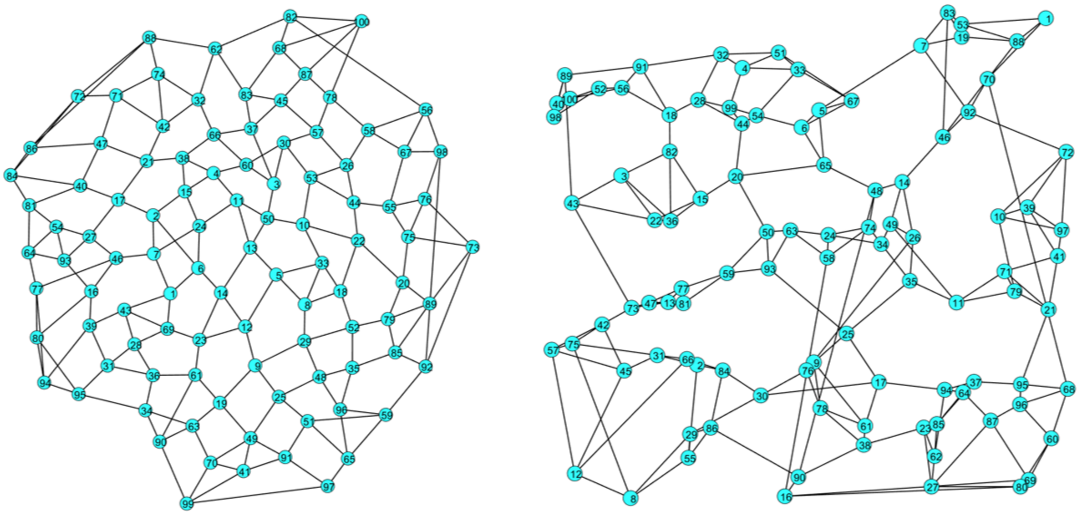

Figure 6.2: A visualization of a *Uniform* instance (left) and *Random* instance (right)

6.4.2 Applying Heuristics to Networks

To be able to compare HGA algorithms with other algorithms to gain an insight about their performance, it is decided to modify two well-known heuristics to PD-Clustering and Fuzzy Clustering. Both algorithms have been discussed in §2.2. These algorithms have two main steps, which is called *location* and *allocation*. In *location* phase, given the membership values, center locations are calculated. In *allocation* phase, given the center locations, membership values are calculated. Location-allocation phases are repeated until cluster center locations do not change through the iterations. As mentioned before, both algorithms are designed for problems on plane. Therefore, we modified the heuristics by adding an approximation step for center locations which is called *Mapping*. There are two different types of mapping: *Vertex Mapping* and *Edge Mapping*. In *Vertex Mapping*, centroid location is moved to the closest vertex on the graph, while in *Edge Mapping*, centroid location is moved to the closest point on the graph, which could be a vertex or edge. Mapping procedure is illustrated in Figure 6.3. In the light of the theoretical results obtained, *Vertex Mapping* is used in M-*PD-Clustering* while *Edge Mapping* is used in M-*Fuzzy C-Means*. Pseudocode of the algorithm is provided as Algorithm 10. It is worth noting that in the computational studies, 10 replications are made with the modified heuristics.

Algorithm 10 Modified *PD-Clustering* and Modified *Fuzzy C-Means* Heuristic

Input: $Graph, numClus$
Output: $Solution$
Apply *PD-Clustering* or *Fuzzy C-Means* (*FCM*) given as Algorithm 3
if PD-Clustering Problem is solved **then**
    Map centroid locations to nearest vertices on the graph
else
    Map centroid locations to nearest point on the graph
end if

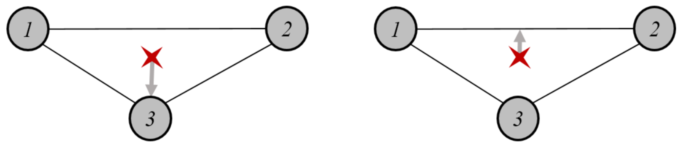

Figure 6.3: *Vertex Mapping*(left) and *Edge Mapping*(right)

6.4.3 Comparison of HGA-N with Modified PD-Clustering Heuristic

Before comparing two solution approaches, it would be helpful to evaluate performances of each heuristic individually. Computational results of HGA-N and modified PD-Clustering Heuristic are given in Appendix C in Tables C.1-C.4, respectively. It is seen that both HGA-N has a robust performance (that is, solution quality is stable within replications) with these problem sets when PD-Clustering Problem is regarded, while the heuristic has higher percentage deviations from the best found solution, which implies that the heuristic is less robust than HGA-N.

To compare outputs of modified PD-Clustering Heuristic and HGA-N for both *Uniform* and *Random* problem sets easier, the Tables 6.10 and 6.11 are provided below. For both *Uniform* and *Random* instances, it is observed that HGA-N performs better than the modified heuristic. On average, solution of the heuristic deviates from that of HGA-N by 2.55% and 4.85% for *Uniform* and *Random* instances, respectively. In the worst case, the heuristic has a 5.68% deviation in *Uniform* instances, and 9.91% in *Random* instances. The heuristic is outperformed by HGA-N more in *Random* instances. In both problem sets, heuristic deviated more as the number of clusters increase. Regarding runtimes, we can say that modified heuristic performs better in overall since the maximum duration for that is approximately 3 seconds. For the smaller instances having up to 100 vertices, HGA-N has lower computing times than M-*PD-Clustering*. For the larger instances, M-*PD-Clustering* has lower runtimes. These observations indicate higher computational complexity, but better solution quality in HGA-N.

6.4.4 Comparison of HGA-E with Modified *Fuzzy C-Means* Heuristic

Similar steps are followed in this part as well. Again, the modified heuristic and HGA are evaluated individually first. In Appendix C, computational results of HGA-E and modified *Fuzzy C-Means* Heuristic are given in Tables C.5, C.7 and C.6, C.8. It could be observed that HGA-E has low deviation values, average of which is less than 1%, while the heuristic has substantially high deviation values. For both problem sets, we can say that HGA-E is more robust than the heuristic.

Tables 6.12 and 6.13 are given to be able to compare HGA-N and the heuristic conveniently. For both *Uniform* and *Random* instances, it is observed that HGA-E outperforms the heuristic in terms of the solution quality. Average deviations from the HGA-E solutions are 7.36% and 12.10% for *Uniform* and random problem sets, respectively. The highest deviations are 37.02% and 47.16% for *Uniform* and *Random* instances, respectively. When deviation values are investigated individually, it could be seen that the values increase as number of clusters increase. However, when runtimes are checked, it could be said that there are significant differences between runtimes of the heuristic and HGA-E. HGA-E has higher runtimes than M-*Fuzzy C-Means*. In short, HGA-E is better in solution quality, whereas the heuristic is better in runtime.

6.5 Center Collision

Center collision is the case when more than one cluster centers are located on the same location. In hard assignment problems, center collision is not observed. Cluster centers cannot collide at optimality in hard assignment problems since network is

Table 6.10: Comparison of HGA-N with M-*PD-Clustering* for PD-Clustering Problem in *Uniform* instances

Instance	Vertices-Clusters-Edges	Best Found Value (HGA-N)	Best Found Value (M-PD)	% Dev from HGA-N	Runtime HGA-N (sec)	Runtime Heur (M-PD)
Unif1	10-3-20	213.26	213.26	-	0.03	2.17
Unif2	10-5-20	87.64	92.62	5.68	0.06	0.74
Unif3	30-2-60	2015.38	2015.82	-	0.05	1.14
Unif4	30-5-60	731.51	745.46	1.91	0.14	0.91
Unif5	30-10-60	284.49	295.89	4.01	0.31	1.28
Unif6	50-2-100	4563.88	4563.88	-	0.07	1.58
Unif7	50-5-100	1583.40	1610.42	1.71	0.26	0.99
Unif8	50-10-100	749.22	783.18	4.53	0.66	0.79
Unif9	100-5-200	4833.39	4921.43	1.82	0.43	0.95
Unif10	100-10-200	2332.16	2429.60	4.18	1.16	0.89
Unif11	100-10-200	2329.87	2425.52	4.11	1.24	0.90
Unif12	100-20-200	1005.43	1050.63	4.50	3.78	1.61
Unif13	100-34-200	480.87	502.69	4.54	9.89	1.07
Unif14	200-5-400	13649.64	13782.08	0.97	0.62	1.49
Unif15	200-10-400	6847.57	6970.27	1.79	2.40	1.13
Unif16	200-20-400	3271.08	3396.60	3.84	8.83	1.10
Unif17	200-40-400	1433.28	1506.83	5.13	23.40	1.43
Unif18	200-67-400	698.51	724.41	3.71	64.96	1.56
Unif19	200-5-800	12681.06	12714.64	-	0.65	1.64
Unif20	200-10-800	6060.36	6159.83	1.64	2.37	1.17
Unif21	200-20-800	2913.24	3020.56	3.68	8.85	1.92
Unif22	200-40-800	1290.65	1326.52	2.78	26.19	1.31
Unif23	200-67-800	653.25	676.47	3.55	52.07	1.65
Unif24	300-5-600	25856.59	26021.06	0.64	0.85	1.76
Unif25	300-10-600	12597.72	12855.36	2.05	3.28	1.38
Unif26	300-30-600	3805.94	3990.97	4.86	20.62	1.38
Unif27	300-60-600	1705.46	1794.60	5.23	67.59	1.89
Unif28	300-100-600	857.14	890.62	3.91	177.67	2.07
Unif29	300-5-1200	23440.09	23475.62	-	0.86	1.77
Unif30	300-10-1200	11066.61	11213.98	1.33	3.20	2.21
Unif31	300-30-1200	3520.18	3640.19	3.41	22.05	1.53
Unif32	300-60-1200	1550.89	1604.06	3.43	74.26	1.75
Unif33	300-100-1200	801.13	826.07	3.11	142.57	2.18
Unif34	300-5-1800	22761.22	22778.70	-	0.93	1.71
Unif35	300-10-1800	11020.74	11144.15	1.12	3.67	2.17
Unif36	300-30-1800	3441.03	3546.49	3.07	27.73	1.59
Unif37	300-60-1800	1562.45	1616.21	3.44	89.06	1.90
Unif38	300-100-1800	793.18	820.02	3.38	171.27	2.43
Unif39	400-5-800	39692.01	40212.43	1.31	1.19	2.45
Unif40	400-10-800	19528.20	19850.44	1.65	4.47	2.58
Unif41	400-40-800	4475.42	4690.10	4.80	46.78	1.93
Unif42	400-80-800	1964.99	2045.70	4.11	142.04	2.33
Unif43	400-134-800	1002.30	1030.18	2.78	344.83	2.77
Unif44	400-5-1600	35381.85	35427.14	-	1.11	2.40
Unif45	400-10-1600	17452.06	17685.34	1.34	4.29	1.97
Unif46	400-40-1600	4040.13	4222.24	4.51	49.79	1.93
Unif47	400-80-1600	1809.82	1877.56	3.74	158.22	2.29
Unif48	400-134-1600	904.75	930.77	2.88	342.23	3.58
Unif49	400-5-2400	35137.15	35164.04	-	1.30	2.37
Unif50	400-10-2400	17124.52	17281.96	0.92	4.80	2.02
Unif51	400-40-2400	3927.64	4057.30	3.30	58.96	2.14
Unif52	400-80-2400	1770.82	1843.66	4.11	172.13	2.50
Unif53	400-134-2400	889.26	909.78	2.31	432.86	3.19
Unif54	400-5-3200	34913.11	34949.88	-	1.30	2.51
Unif55	400-10-3200	17183.09	17332.06	0.87	5.52	2.12
Unif56	400-40-3200	3937.07	4073.71	3.47	62.04	2.16
Unif57	400-80-3200	1776.13	1837.27	3.44	201.35	2.58
Unif58	400-134-3200	887.91	914.74	3.02	436.90	3.20
Unif59	600-5-1200	71662.16	72079.02	0.58	1.70	3.26
Unif60	600-10-2400	31852.59	32020.83	0.53	7.43	2.49
		Average		**2.55**	**58.25**	**1.86**

Table 6.11: Comparison of HGA-N with M-*PD-Clustering* for PD-Clustering Problem in *Random* instances

Instance	Vertices-Clusters-Edges	Best Found Value (HGA-N)	Best Found Value (Heur)	% Dev from HGA-N	Runtime HGA-N (sec)	Runtime M-PD (sec)
Rand1	10-3-20	171.79	171.79	-	0.04	0.86
Rand2	10-5-20	71.85	74.65	3.90	0.05	1.18
Rand3	30-2-60	1652.45	1683.36	1.87	0.05	0.74
Rand4	30-5-60	551.51	556.34	0.87	0.16	0.73
Rand5	30-10-60	245.42	266.17	8.45	0.37	0.85
Rand6	50-2-100	3958.35	3966.10	-	0.06	1.34
Rand7	50-5-100	1449.71	1529.81	5.53	0.28	0.97
Rand8	50-10-100	592.15	637.65	7.68	0.66	1.60
Rand9	100-5-200	3917.06	4008.95	2.35	0.36	1.42
Rand10	100-10-200	1789.41	1896.52	5.99	1.09	0.98
Rand11	100-10-200	1860.13	1967.49	5.77	1.18	0.97
Rand12	100-20-200	788.38	863.73	9.56	3.46	0.94
Rand13	100-34-200	393.33	424.05	7.81	12.41	1.00
Rand14	200-5-400	12054.54	12441.54	3.21	0.61	2.01
Rand15	200-10-400	5879.24	6287.66	6.95	1.98	1.37
Rand16	200-20-400	2645.00	2809.87	6.23	7.56	1.10
Rand17	200-40-400	1146.90	1231.55	7.38	26.04	1.22
Rand18	200-67-400	553.17	607.98	9.91	69.07	1.61
Rand19	200-5-800	10470.84	10639.96	1.62	0.66	2.22
Rand20	200-10-800	4865.25	4974.89	2.25	1.97	1.97
Rand21	200-20-800	2172.54	2289.88	5.40	8.63	1.58
Rand22	200-40-800	1003.63	1080.13	7.62	28.97	1.33
Rand23	200-67-800	508.63	549.46	8.03	77.85	2.03
Rand24	300-5-600	22797.94	23244.76	1.96	0.82	2.00
Rand25	300-10-600	10927.29	11467.04	4.94	2.92	2.09
Rand26	300-30-600	3253.58	3531.87	8.55	21.58	1.47
Rand27	300-60-600	1400.81	1520.18	8.52	85.72	2.16
Rand28	300-100-600	709.90	762.14	7.36	231.47	2.33
Rand29	300-5-1200	19002.21	19055.84	-	0.91	2.16
Rand30	300-10-1200	9021.96	9308.12	3.17	2.96	2.30
Rand31	300-30-1200	2772.83	3004.32	8.35	20.96	1.55
Rand32	300-60-1200	1242.36	1332.47	7.25	87.21	2.42
Rand33	300-100-1200	623.10	661.02	6.09	242.33	2.10
Rand34	300-5-1800	18587.80	18659.37	-	0.88	2.39
Rand35	300-10-1800	8744.83	9004.01	2.96	2.92	2.09
Rand36	300-30-1800	2657.48	2840.51	6.89	22.67	1.65
Rand37	300-60-1800	1179.35	1252.51	6.20	95.47	1.80
Rand38	300-100-1800	582.16	616.20	5.85	290.25	2.17
Rand39	400-5-800	35321.72	35810.02	1.38	1.00	3.01
Rand40	400-10-800	16440.96	17292.16	5.18	4.34	2.30
Rand41	400-40-800	3715.01	3962.43	6.66	47.61	1.99
Rand42	400-80-800	1696.47	1823.37	7.48	163.95	2.32
Rand43	400-134-800	844.87	901.66	6.72	518.92	2.84
Rand44	400-5-1600	29290.89	29548.37	0.88	1.14	3.23
Rand45	400-10-1600	13914.49	14349.68	3.13	3.75	2.43
Rand46	400-40-1600	3150.73	3356.87	6.54	49.97	2.17
Rand47	400-80-1600	1401.28	1497.81	6.89	187.00	2.32
Rand48	400-134-1600	708.22	752.78	6.29	551.02	2.94
Rand49	400-5-2400	29222.82	29359.89	-	1.13	3.23
Rand50	400-10-2400	13828.19	14080.58	1.83	4.91	2.58
Rand51	400-40-2400	3091.03	3283.23	6.22	53.21	2.24
Rand52	400-80-2400	1364.23	1453.68	6.56	209.39	3.32
Rand53	400-134-2400	689.15	727.97	5.63	662.51	2.99
Rand54	400-5-3200	28469.36	28506.42	-	1.33	3.14
Rand55	400-10-3200	13481.58	13842.32	2.68	4.62	3.26
Rand56	400-40-3200	3053.87	3228.46	5.72	62.89	2.25
Rand57	400-80-3200	1367.72	1453.16	6.25	221.90	2.73
Rand58	400-134-3200	669.21	697.31	4.20	788.17	3.25
Rand59	600-5-1200	64906.87	66522.30	2.49	1.73	4.54
Rand60	600-10-2400	26152.89	26638.87	1.86	7.36	4.27
		Average		**4.85**	**81.67**	**2.07**

Table 6.12: Comparison of HGA-E with the Heuristic for Fuzzy Clustering Problem in *Uniform* instances

Instance	Vertices-Clusters-Edges	Best Found Value (HGA-E)	Best Found Value (M-FCM)	% Dev from HGA-N	Runtime HGA-E (sec)	Runtime M-FCM (sec)
Unif1	10-3-20	6655.58	8859.17	33.11	1.24	1.05
Unif2	10-5-20	1556.71	1617.13	3.88	2.21	0.86
Unif3	30-2-60	160131.33	167675.92	4.71	1.16	1.80
Unif4	30-5-60	22152.84	24222.26	9.34	5.78	1.51
Unif5	30-10-60	4125.11	4533.42	9.90	15.53	1.11
Unif6	50-2-100	479214.10	527752.67	10.13	1.62	2.27
Unif7	50-5-100	59611.87	62723.42	5.22	6.89	2.85
Unif8	50-10-100	14451.55	14978.02	3.64	20.57	1.45
Unif9	100-5-200	265868.13	287149.10	8.00	11.72	3.09
Unif10	100-10-200	63812.88	66531.01	4.26	37.03	2.41
Unif11	100-10-200	63827.80	66860.66	4.75	35.67	2.02
Unif12	100-20-200	13116.98	13536.41	3.20	123.26	1.82
Unif13	100-34-200	3623.30	3889.82	7.36	297.78	2.25
Unif14	200-5-400	1047129.21	1151522.07	9.97	23.28	6.75
Unif15	200-10-400	262448.37	272463.37	3.82	77.73	5.69
Unif16	200-20-400	62079.05	65507.49	5.52	232.48	2.92
Unif17	200-40-400	13425.24	14286.28	6.41	719.78	4.01
Unif18	200-67-400	3818.47	4018.41	5.24	1695.89	5.46
Unif19	200-5-800	892697.57	1184062.22	32.64	64.03	4.81
Unif20	200-10-800	205094.98	212317.08	3.52	244.18	5.15
Unif21	200-20-800	50154.47	52140.66	3.96	895.96	4.07
Unif22	200-40-800	10919.94	11432.29	4.69	3021.46	5.85
Unif23	200-67-800	3318.67	3453.65	4.07	8135.81	8.69
Unif24	300-5-600	2474648.85	2647378.07	6.98	33.74	7.88
Unif25	300-10-600	586791.56	610258.39	4.00	124.30	7.94
Unif26	300-30-600	56589.18	59948.24	5.94	614.87	4.89
Unif27	300-60-600	12720.11	13567.98	6.67	2168.65	7.08
Unif28	300-100-600	3842.52	4096.60	6.61	6035.54	10.55
Unif29	300-5-1200	2039803.05	2160966.72	5.94	109.03	18.04
Unif30	300-10-1200	453307.97	468558.73	3.36	393.55	9.19
Unif31	300-30-1200	48435.21	50623.83	4.52	2692.02	7.03
Unif32	300-60-1200	10432.58	11056.87	5.98	9738.68	11.45
Unif33	300-100-1200	3314.90	3471.09	4.71	25134.16	18.05
Unif34	300-5-1800	1917230.46	2626982.41	37.02	192.72	8.53
Unif35	300-10-1800	448921.85	466590.34	3.94	836.54	11.79
Unif36	300-30-1800	46281.74	49218.22	6.34	5528.38	9.94
Unif37	300-60-1800	10577.11	11123.44	5.17	19125.57	16.14
Unif38	300-100-1800	3252.36	3480.85	7.03	52921.46	25.18
Unif39	400-5-800	4324419.85	4556817.00	5.37	42.30	8.37
Unif40	400-10-800	1057090.18	1096703.44	3.75	153.90	8.77
Unif41	400-40-800	58531.25	62438.41	6.68	1405.21	7.46
Unif42	400-80-800	12675.22	13586.69	7.19	4687.63	11.70
Unif43	400-134-800	3928.98	4119.31	4.84	13600.78	17.78
Unif44	400-5-1600	3475104.54	3724559.20	7.18	144.65	13.12
Unif45	400-10-1600	838938.46	867975.34	3.46	595.97	11.43
Unif46	400-40-1600	48077.07	50621.80	5.29	6141.21	11.49
Unif47	400-80-1600	10699.23	11243.61	5.09	20966.77	19.23
Unif48	400-134-1600	3178.91	3347.03	5.29	58307.18	30.43
Unif49	400-5-2400	3417234.38	4119848.02	20.56	269.76	11.23
Unif50	400-10-2400	806814.99	831725.77	3.09	1296.82	13.20
Unif51	400-40-2400	45121.41	47428.94	5.11	12575.83	15.64
Unif52	400-80-2400	10233.99	10843.78	5.96	44332.12	26.92
Unif53	400-134-2400	3065.18	3231.12	5.41	132329.73	42.98
Unif54	400-5-3200	3358594.45	3889626.56	15.81	453.56	10.35
Unif55	400-10-3200	812646.65	850686.80	4.68	1902.82	14.00
Unif56	400-40-3200	45363.98	47511.76	4.73	20757.00	19.46
Unif57	400-80-3200	10347.27	11039.63	6.69	72079.54	34.65
Unif58	400-134-3200	3054.60	3222.96	5.51	213070.67	56.58
Unif59	600-5-1200	9476422.09	9882690.88	4.29	68.11	23.20
Unif60	600-10-2400	1853226.67	1931256.14	4.21	1033.24	20.29
		Average		**7.36**	**12458.92**	**11.33**

Table 6.13: Comparison of HGA-E with the Heuristic for Fuzzy Clustering Problem in *Random* instances

Instance	Vertices-Clusters-Edges	Best Found Value (HGA-E)	Best Found Value (M-FCM)	% Dev from HGA-N	Runtime HGA-E (sec)	Runtime M-FCM (sec)
Rand1	10-3-20	4322.92	4883.51	12.97	0.81	1.03
Rand2	10-5-20	1047.41	1090.97	4.16	1.65	0.90
Rand3	30-2-60	114353.98	133065.31	16.36	1.14	1.49
Rand4	30-5-60	13461.07	14554.77	8.12	4.84	1.44
Rand5	30-10-60	3095.19	3246.72	4.90	16.66	1.21
Rand6	50-2-100	381819.64	401552.97	5.17	1.51	1.36
Rand7	50-5-100	49577.60	56225.44	13.41	7.13	1.80
Rand8	50-10-100	9118.79	10229.48	12.18	22.21	1.87
Rand9	100-5-200	184413.36	210621.58	14.21	9.67	2.36
Rand10	100-10-200	38521.14	41300.07	7.21	34.58	2.26
Rand11	100-10-200	41668.27	45885.99	10.12	34.34	2.80
Rand12	100-20-200	8154.10	9289.36	13.92	120.08	2.26
Rand13	100-34-200	2407.44	2847.95	18.30	323.36	2.39
Rand14	200-5-400	832510.89	874897.08	5.09	17.35	2.64
Rand15	200-10-400	196558.92	215569.60	9.67	58.09	3.18
Rand16	200-20-400	42076.19	46605.49	10.76	178.97	3.71
Rand17	200-40-400	8677.84	10260.89	18.24	759.00	5.21
Rand18	200-67-400	2421.18	2837.36	17.19	1759.47	5.59
Rand19	200-5-800	619974.72	681907.89	9.99	47.73	3.23
Rand20	200-10-800	137142.01	148680.84	8.41	168.38	4.01
Rand21	200-20-800	28233.05	31414.08	11.27	715.10	4.87
Rand22	200-40-800	6582.09	7302.13	10.94	2403.86	6.98
Rand23	200-67-800	2004.43	2296.36	14.56	6167.42	8.78
Rand24	300-5-600	1950729.78	2090362.18	7.16	26.13	8.57
Rand25	300-10-600	442659.30	476489.76	7.64	95.96	6.47
Rand26	300-30-600	41926.55	47529.67	13.36	581.83	6.87
Rand27	300-60-600	8574.09	9700.77	13.14	1927.61	7.79
Rand28	300-100-600	2658.14	2939.21	10.57	5060.00	10.96
Rand29	300-5-1200	1360727.48	1509220.88	10.91	76.33	4.93
Rand30	300-10-1200	306503.85	321648.47	4.94	273.04	6.65
Rand31	300-30-1200	30125.46	33756.87	12.05	1818.63	9.40
Rand32	300-60-1200	6738.31	7531.33	11.77	7005.78	12.78
Rand33	300-100-1200	2003.18	2236.38	11.64	18828.63	17.84
Rand34	300-5-1800	1288078.96	1419247.99	10.18	131.75	4.98
Rand35	300-10-1800	289619.10	322997.93	11.53	530.42	8.54
Rand36	300-30-1800	27731.79	31735.50	14.44	3583.54	10.95
Rand37	300-60-1800	6034.42	6813.22	12.91	13862.36	16.18
Rand38	300-100-1800	1748.24	1963.89	12.34	39138.58	24.89
Rand39	400-5-800	3463718.14	4541066.99	31.10	33.62	5.32
Rand40	400-10-800	760198.32	827039.69	8.79	108.11	8.23
Rand41	400-40-800	40799.08	45811.72	12.29	1181.04	10.27
Rand42	400-80-800	9434.29	10996.01	16.55	3998.40	12.56
Rand43	400-134-800	2801.27	3141.39	12.14	11214.83	18.35
Rand44	400-5-1600	2383757.76	2601985.64	9.15	83.24	5.38
Rand45	400-10-1600	548209.44	593105.39	8.19	377.79	6.89
Rand46	400-40-1600	29387.36	33513.46	14.04	4257.15	13.61
Rand47	400-80-1600	6381.86	7171.00	12.37	14259.10	19.86
Rand48	400-134-1600	1949.63	2175.27	11.57	42790.46	30.21
Rand49	400-5-2400	2375468.90	2679823.06	12.81	172.30	6.19
Rand50	400-10-2400	530390.82	568011.53	7.09	696.92	8.28
Rand51	400-40-2400	28059.38	32104.86	14.42	8072.04	16.97
Rand52	400-80-2400	6049.09	6726.87	11.20	28975.33	27.46
Rand53	400-134-2400	1837.80	2038.77	10.94	88997.17	42.71
Rand54	400-5-3200	2232532.18	3285406.89	47.16	274.18	7.34
Rand55	400-10-3200	513240.35	547134.08	6.60	1350.56	9.37
Rand56	400-40-3200	27280.05	30788.64	12.86	13002.71	22.63
Rand57	400-80-3200	6053.57	6867.96	13.45	49737.58	35.64
Rand58	400-134-3200	1730.19	1906.36	10.18	154880.69	55.71
Rand59	600-5-1200	7642367.47	8772882.86	14.79	62.36	6.73
Rand60	600-10-2400	1260072.74	1339688.91	6.32	706.25	12.45
		Average		**12.10**	**8850.43**	**10.19**

partitioned in the hard clustering problem. However, in soft assignment problems no partitioning occurs (that is, every vertex is assigned to every cluster), center collision could be observed. In this section, center collision in soft assignment problems will be described using two example problems.

The first example is a network with the shape of polygon. Each corner and center of the polygon corresponds to one vertex. The network with the shape of pentagon is illustrated in Figure 6.4. We can have different examples with different number of sides. As another example, we may take an octagon (which corresponds to a network with 9 vertices) and solve all the four types of clustering problems with HGA-N. Figure 6.5 presents the solutions for four types of problems for an octagon network with 2 clusters. Cluster centers are shown in bold in the figure. The hard clustering solutions are given in bottom cells. For P-Median Problem, cluster centers are found as vertices 2 and 9. Vertices are colored according to their assignment to clusters. For example, vertices 1, 2 and 3 are assigned to the cluster with center that is on vertex 2, and the rest is assigned to the other cluster. For SSC Problem, cluster centers are found as vertices 1 and 9. In this example, solutions of P-Median Problem and Sum of Squares Clustering Problem are symmetrical. In none of these two, center collision is observed. However, if PD-Clustering Problem is solved with the same network, both centers are located to vertex 9. For Fuzzy Clustering with fuzzifier constant m=3, both centers are found as vertex 9 again. In short, in this example, in both soft clustering problems, cluster centers collide.

The second example is a symmetrical network with two vertices in the middle that are connected, and a number of vertices connected to the vertices in the middle. We call this example an *H-tree*. An H-tree network example is given in the Figure 6.6. In H-tree, there are $2n + 2$ vertices in total and n vertices are connected to each vertex in the center. Each edge has a length of 1 unit. Both PD-Clustering Problem and Fuzzy Clustering Problem with m=3 are solved for H-tree networks with different n values, and with 3 clusters. The solutions of HGA-N are illustrated in Figure 6.7. The vertices selected as cluster centers are shown with bold lines. When n=2, in PD-Clustering Problem, cluster centers are vertices 1, 2 and 4, while these are vertices 3 and 4 for Fuzzy Clustering Problem. So for n=2, centers collide in Fuzzy Clustering Problem. When n=3, as in n=2, centers collide in Fuzzy Clustering Problem. When n=4, in

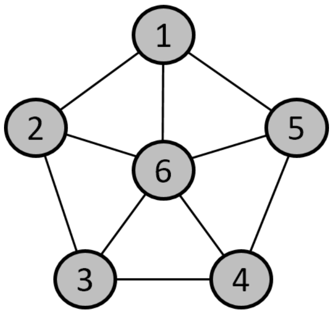

Figure 6.4: A network example with the shape of pentagon

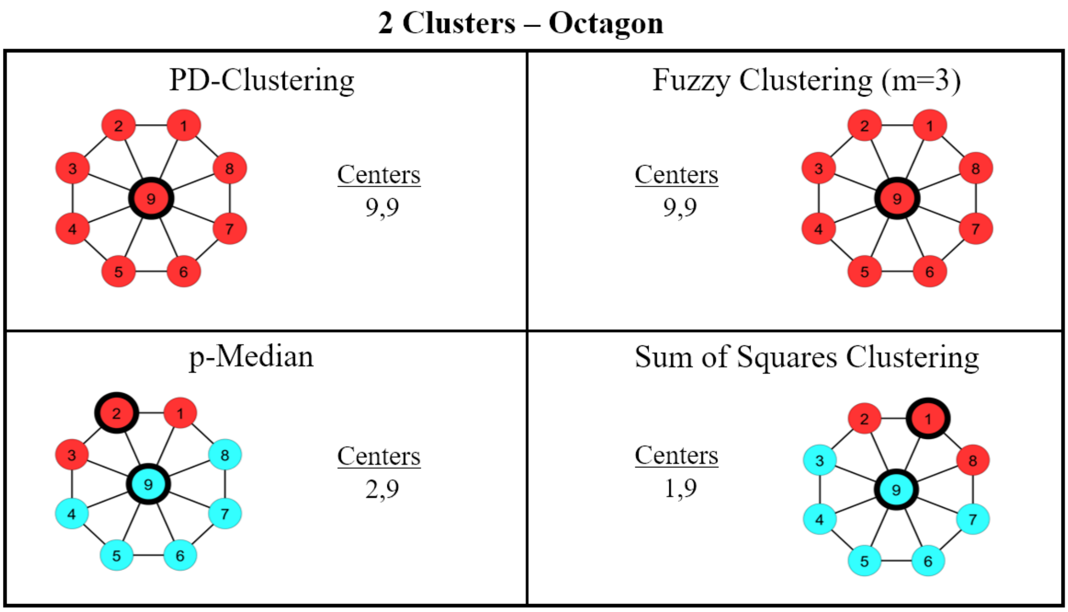

Figure 6.5: Solutions to a network with the shape of octagon when there are 2 clusters

both PD-Clustering and Fuzzy Clustering Problem, centers collide. With these three n values, we observed that as n increases, centers start to collide. In Appendix A, proof of this observation is provided.

In brief, in soft clustering problems, more than one center can be located on a vertex. This phenomenon is also observed in different problem instances that are solved.

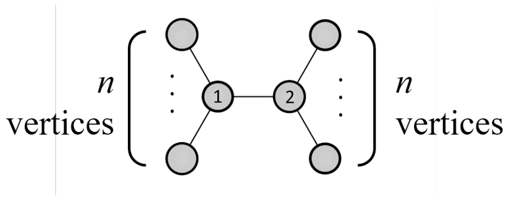

Figure 6.6: An H-tree network with *n* vertices connected to both vertices in the middle

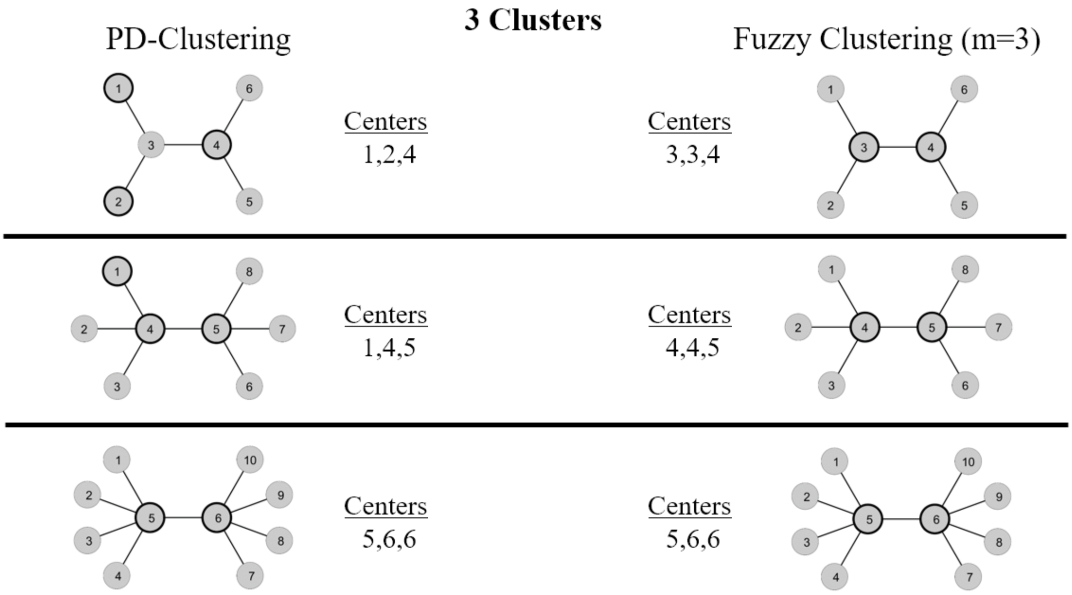

Figure 6.7: Solutions for H-tree with different n values

6.5.1 Comparison of Center Locations

As four different clustering problems are solved and analyzed for the ORLib dataset, analyzing differences in solutions is considered as important to gain insight about these problems. The solutions for selected problem instances are compared, and number of common centers are counted. The results are given in Table 6.14 where each column represents the results for every pair of clustering problems. Looking at the first column **p-med & PD** in which P-Median and PD-Clustering solutions are compared, we can see that there are similarities between their solutions, but the highest similarity observed is 50% which is in ORLib 3, 12, 32 and 36 instances. As a common feature, all these instances have 10 clusters. In the instances with 5 clusters, the highest similarity is 60% which is observed in ORLib 6, 21 and 35 instances. In the second column, SSC and Fuzzy Clustering Problem are compared. The highest similarity with 10 cluster instances and 5 cluster instances are 50% and 60%, respectively. When PD-Clustering and fuzzy clustering solutions are compared (given in the third column), it could be noted that both clustering algorithms find exactly the same centers in 7 instances. Based on the average values, we can say that number of common centers with 5 clusters are slightly higher than these with 10 clusters in every pair of comparisons. Additionally, it could be seen that the highest similarity is in the column **PD & Fuzzy**. To sum up, different clustering approaches may produce common centers on the selected instances (up to a point). It is seen that similarity of the solutions depends on the assignment type. That is, higher similarity in solutions between the clustering approaches attained when their assignment types are the same.

6.5.2 Center Collision in ORLib Instances

As discussed in §6.5, center collision is observed in soft assignment problems, that is, more than one cluster centers could be located to the same location. In Table 6.15, number of centers collide in PD-Clustering and fuzzy clustering is reported. If there is no collision, it is shown with "-" symbol. In the case of collisions, number of centers collide on vertices are reported. For example, in a solution with centers on vertices $4, 7, 7, 13, 13, 13, 92$, there are collisions in two different places. Therefore,

Table 6.14: Comparison of center locations for different problems

Instance	Vertices-Clusters-Edges	Number of Common Centers (p-med & PD)	(SSC & Fuzzy)	(PD & Fuzzy)
ORLib1	100-5-198	-	2	2
ORLib2	100-10-193	2	1	7
ORLib3	100-10-198	5	5	8
ORLib6	200-5-786	3	1	5
ORLib7	200-10-779	2	3	9
ORLib11	300-5-1772	1	1	4
ORLib12	300-10-1758	5	5	9
ORLib16	400-5-3153	1	1	5
ORLib17	400-10-3142	2	3	5
ORLib21	500-5-4909	3	3	5
ORLib22	500-10-4896	4	4	10
ORLib26	600-5-7069	2	2	5
ORLib27	600-10-7072	1	2	9
ORLib31	700-5-9601	3	2	5
ORLib32	700-10-9584	5	5	8
ORLib35	800-5-12548	3	3	5
ORLib36	800-10-12560	5	5	9
ORLib38	900-5-15898	1	2	3
ORLib39	900-10-15896	2	4	8
Average		2.63	2.84	6.37
	5 Clusters	1.89	1.89	4.33
	10 Clusters	3.30	3.70	8.20

the reported result will be "$2, 3$", since we have two centers in vertex 7 and three centers in vertex 13. Looking at the results, there is no collision in 2 instances and 1 instance in PD-Clustering and fuzzy clustering, respectively. The highest collision occurred in ORLib7, in which 6 and 7 centers are located to the same vertex in PD-Clustering and Fuzzy Clustering solutions. In overall, it is observed that the number of centers collide in Fuzzy Clustering is more than that in PD-Clustering Problem.

Table 6.15: Number of centers collide for selected ORLib instances

Instance	Vertices-Clusters-Edges	Number of Centers Collide	
		PD-Clustering	Fuzzy Clustering
ORLib1	100-5-198	2	2,2
ORLib2	100-10-193	3	5
ORLib3	100-10-198	2	2
ORLib6	200-5-786		
ORLib7	200-10-779	6	7
ORLib11	300-5-1772	2	2,2
ORLib12	300-10-1758	2	3
ORLib16	400-5-3153	2	2
ORLib17	400-10-3142	2,4	2,4
ORLib21	500-5-4909	2,2	2,2
ORLib22	500-10-4896	3	3
ORLib26	600-5-7069	2	2
ORLib27	600-10-7072	6,2	2,6,2
ORLib31	700-5-9601	2,2	2,2
ORLib32	700-10-9584		2
ORLib35	800-5-12548	2	2
ORLib36	800-10-12560	2	2,2
ORLib38	900-5-15898	2	2
ORLib39	900-10-15896	2,6	2,5

CHAPTER 7

CONCLUSION

In this thesis, Center-Based Clustering Problems on Networks have been analyzed. In the scope of this study, the following four problems have been investigated that differ in assignment scheme and objective function used.

- P-Median Problem,
- Sum of Squares Clustering Problem,
- PD-Clustering Problem,
- Fuzzy Clustering Problem.

Among these problems, P-Median Problem is a well-known Facility Location Problem. Planar case of Sum of Squares Clustering Problem is also widely studied in the literature, and network version of the problem is studied in [1]. Besides these two hard assignment problems, two clustering problems on networks that use soft assignment scheme are newly studied: PD-Clustering Problem and Fuzzy Clustering Problem.

In order to analyze these problems on hand, a framework is used. This framework is inspired by the studies [28], [22], [23] and [24] which mainly focus on finding theoretical properties of optimal solutions of certain Facility Location Problems. In our case, we analyzed these clustering problems in order to derive theoretical results. We prove that the optimal cluster centers are always located on vertices $\mathbf{V}$ in PD-Clustering Problem. For Fuzzy Clustering Problem, we found that the optimal centers could be located anywhere on the network $\mathbf{G} = (\mathbf{E}, \mathbf{V})$. Summarizing the results, it is found that cluster centers will be located on $\mathbf{V}$ in P-Median Problem and PD-Clustering Problem while these could be located anywhere on the network in Sum of Squares Clustering Problem and Fuzzy Clustering Problem.

With the derived results, a solution framework is developed which is a Genetic Algorithm with Local Search embedded. This solution approach is called Hybrid Genetic Algorithm (HGA). Benefiting the theoretical results obtained, two versions of HGA are proposed: Node Based HGA (HGA-N) and Edge Based HGA (HGA-E). HGA-N is proposed for P-Median Problem and PD-Clustering Problem, and HGA-E is designed for Sum of Squares Clustering Problem and Fuzzy Clustering Problem. In order to test performance of these algorithms, if available, benchmark instances are solved, and the solutions available in the literature are used. For P-Median Problem, HGA-N is able to find solutions with insignificant percentage deviations from the reported optimal objective function values. For Sum of Squares Clustering Problem, HGA-E is able to find solutions that have lower objective function values than the ones reported in [1]. Since PD-Clustering Problem and Fuzzy Clustering Problem problems are newly studied on networks, we do not have previously reported objective values for these. Therefore, heuristics that are well-known for the planar versions of these problems are modified for the network case. Since these heuristics require vertex coordinates, two data sets (*Uniform* and *Random*) are generated by two different procedures. Compared to these heuristics, HGA-N and HGA-E find considerably better solutions for PD-Clustering Problem and Fuzzy Clustering Problem, respectively. It could be concluded that HGA has a promising performance for the problems on hand.

This study has three main theoretical contributions. First, to the best of our knowledge, soft clustering problems are newly studied by us. Second, Center-Based Clustering Problems have been analyzed from a Location Theory perspective, and theoretical results are obtained for all the problems studied. For PD-Clustering Problem, it is found that the optimal solution is on vertices. In other words, regardless of the assignment scheme, in two clustering problems (P-Median Problem and PD-Clustering Problem) that use sum of distance as objective function, cluster centers will be located on vertices. Third, a solution framework has been developed for these problems that is called HGA. HGA-N finds solutions objective function of which is very close to the ones reported, and HGA-E outperformed results reported in the literature.

This study could be utilized in developing approaches to problems from various disciplines, such as Sensor Networks, Emergency Medical Services (EMS) Location Problems, Protein-Protein Interaction (PPI) Problems and Humanitarian Logistics Problems.

A future research direction could be defining new problems with different objectives, such as an objective function as a survival function that decreases with the distance. A limitation of this study is that it is defined on Euclidean Graphs, that is, a network satisfying metric properties. Other types of networks that do not satisfy metric properties could be studied in the future, such as social networks. Furthermore, there are studies in the literature to map networks that do not satisfy Euclidean properties to Euclidean networks. With the help of these approaches, application areas of this study could be extended further.

REFERENCES

[1] E. Carrizosa, N. Mladenovic, and R. Todosijevic, "Variable neighborhood search for minimum sum-of-squares clustering on networks," *European Journal of Operational Research*, vol. 230, pp. 356,363, 2013.

[2] J. A. Hartigan, *Clustering Algorithms*. John Wiley and Sons, 1975.

[3] S. Ayramo and T. Karkkainen, "Introduction to partitioning based clustering methods with a robust example," *Report*, 2006.

[4] S. E. Schaeffer, "Graph clustering," *Computer Science Review I*, pp. 27–64, 2007.

[5] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: a review," *ACM computing surveys (CSUR)*, vol. 31, no. 3, pp. 264–323, 1999.

[6] P. Berkhin, "A survey of clustering data mining techniques," in *Grouping multidimensional data*, pp. 25–71, Springer, 2006.

[7] R. Xu and D. Wunsch, "Survey of clustering algorithms," *IEEE Transactions on neural networks*, vol. 16, no. 3, pp. 645–678, 2005.

[8] S. C. Johnson, "Hierarchical clustering schemes," *Psychometrika*, vol. 32, no. 3, pp. 241–254, 1967.

[9] J. Hartigan and M. A. Wong, "A k-means clustering algorithm," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 28, no. 1, pp. 100–108, 1979.

[10] L. Kaufman and P. J. Rousseeuw, *Finding Groups in Data: an Introduction to Cluster Analysis*. John Wiley and Sons, 2005.

[11] H. J. Zimmermann, "Fuzzy set theory," *WIREs Computational Statistics*, vol. 2, 2010.

[12] J. C. Bezdek, R. Ehrlich, and W. Full, "Fcm: The fuzzy c-means clustering algorithm," *Computers and Geosciences*, vol. 10, no. 2-3, pp. 191–203, 1984.

[13] C. Iyigun and A. Ben-Israel, "Probabilistic d-clustering," *Journal of Classification*, vol. 25, pp. 5–26, 2008.

[14] J. H. Holland, *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.

[15] K. Krishna and M. Narasimha Murty, "Genetic k-means algorithm," *IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics*, vol. 29, no. 3, 1999.

[16] W. Sheng and X. Liu, "A genetic k-medoids clustering algorithm," *Journal of Heuristics*, vol. 12, pp. 447–466, 2006.

[17] C. C. Aggarwal and H. Wang, *Managing and Mining Graph Data*, ch. A Survey of Clustering Algorithms for Graph Data. Springer Science+Business Media, 2010.

[18] E. L. Johnson, A. Mehrotra, and G. L. Nemhauser, "Min-cut clustering," *Mathematical Programming*, vol. 62, pp. 133–151, 1993.

[19] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, 1993.

[20] B. W. Kerninghan and S. Lin, "An efficient heuristic procedure for partitioning graphs," *The Bell System Technical Journal*, pp. 291–307, 1970.

[21] M. E. J. Newman and M. Girvan, "Community structure in social and biological networks," *Proceedings of the National Academy of Sciences*, vol. 99, pp. 7821–7826, 2002.

[22] S. Hakimi, "Optimum locations of switching centers and the absolute centers and medians of a graph," *Operations Research*, vol. 12, pp. 450–459, June 1964.

[23] S. L. Hakimi, "Optimum distribution of switching centers in a communication network and some related graph theoretic problems," *Operational Research*, vol. 13, pp. 462–475, 1965.

[24] J. Levy, “An extended theorem for location on a network,” *Operational Research Quarterly*, vol. 18, no. 4, pp. 433–442, 1967.

[25] N. Mladenovic, J. Brimberg, P. Hansen, and J. A. Moreno-Perez, “The p-median problem: A survey of metaheuristic approaches,” *European Journal of Operations Research*, vol. 179, pp. 927–939, 2007.

[26] C. M. Hosage and M. F. Goodchild, “Discrete space location-allocation solutions from genetic algorithms,” *Annals of Operations Research*, vol. 6, pp. 35–46, 1986.

[27] O. Alp, E. Erkut, and Z. Drezner, “An efficient genetic algorithm for the p-median problem,” *Annals of Operations Research*, vol. 122, pp. 21–42, 2003.

[28] J. N. Hooker, R. S. Garfinkel, and C. K. Chen, “Finite dominating sets for network location problems,” *Operations Research*, vol. 39, no. 1, 1991.

[29] C. Iyigun and A. Ben-Israel, *Probabilistic distance clustering*. PhD thesis, Rutgers, The State University of New Jersey, 2007.

[30] M. Lozano and C. G. Martinez, “Hybrid metaheuristics with evolutionary algorithms specializing in intensification and diversification: Overview and progress report,” *Computers & Operations Research*, vol. 37, pp. 481–497, 2010.

[31] C. G. Martinez and M. Lozano, *Local Search Based on Genetic Algorithms*, pp. 199–222. 2008.

[32] R. C. Prim, “Shortest connection networks and some generalizations,” *Bell system technical journal*, vol. 36, no. 6, pp. 1389–1401, 1957.

Appendix A

CENTER COLLISION ON AN H-TREE GRAPH

In this chapter, center collision will be illustrated on an H-Tree graph example previously introduced in Chapter 6. In this particular example, for simplicity, we assume that each edge has a length of 1. There are two central vertices connected to each other. And each central vertex has n vertices connected, additional to other central vertex. It will be shown that when we solve both PD-Clustering and Fuzzy Clustering problems for 3 clusters on this graph, center collision will occur as n increases.

The H-Tree graph for this example is given in Figure A.1. For the sake of simplicity, two vertex sets are created: V_{left} and V_{right}, each of which contains n vertices. For the case of 3 clusters, the following cases could be observed regarding the locations of cluster centers (cases that are the same due to symmetry are listed together).

1. $\{v_1, v_1, v_1\}$ or $\{v_2, v_2, v_2\}$
2. $\{v_1, v_1, v_2\}$ or $\{v_1, v_2, v_2\}$
3. $\{v_i,\ v_j,\ v_k,\ i \neq j \neq k \in V_{left}\}$ or $\{v_i,\ v_j,\ v_k,\ i \neq j \neq k \in V_{right}\}$

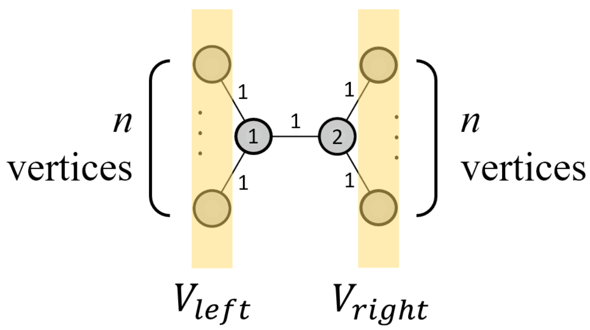

Figure A.1: H-Tree Graph

4. $\{\{v_i,\, i \in V_{left}\}, v_1, v_2\}$ or $\{v_1, v_2, \{v_i,\, i \in V_{right}\}\}$

5. $\{\{v_i,\, v_j,\, i \neq j \in V_{left}\}, v_1\}$ or $\{v_2, \{v_i,\, v_j,\, i \neq j \in V_{right}\}\}$

6. $\{\{v_i,\, v_j,\, i \neq j \in V_{left}\}, v_2\}$ or $\{v_1, \{v_i,\, v_j,\, i \neq j \in V_{right}\}\}$

Since the lengths of all edges are the same, contributions of vertices in V_{left} to the objective functions will be identical. Similarly, vertices in V_{right} will contribute to the objective function equally. By using this property, and taking n as a variable, objective function value could be calculated easily. With the 6 cases given, if we calculate the objective function, we will obtain objective function values which are given in Tables A.1 and A.2.

Table A.1: Objective function values of the given cases for PD-Clustering Problem

	PD-Clustering		
Case	$\mathbf{V_{left} + V_{right}}$	$\mathbf{v_1 + v_2}$	**Total**
1	$n/3 + 2n/3$	$1/3$	$n + 1/3$
2	$2n/5 + n/2$	0	$9n/10$
3	$2(n-3)/3 + n$	1	$(5n-6)/3 + 1$
4	$(n-1)/2 + 6n/11$	0	$(23n-11)/22$
5	$(n-2)/2 + 6n/7$	$1/2$	$(19n-14)/14 + 1/2$
6	$2(n-2)/3 + n/2$	$1/3$	$(7n-8)/6 + 1/3$

Table A.2: Objective function values of the given cases for Fuzzy Clustering Problem

	Fuzzy Clustering (m=3)		
Case	$\mathbf{V_{left} + V_{right}}$	$\mathbf{v_1 + v_2}$	**Total**
1	$n/9 + 4n/9$	$1/9$	$5n/9 + 1/9$
2	$4n/25 + n/4$	0	$41n/100$
3	$4(n-3)/9 + n$	$5/9$	$(13n-12)/9 + 5/9$
4	$(n-1)/4 + 36n/121$	0	$(n-1)/4 + 36n/121$
5	$(n-2)/4 + 36n/49$	$1/4$	$(n-2)/4 + 36n/49 + 1/4$
6	$4(n-2)/9 + n/4$	$1/9$	$(25n-32)/36 + 1/9$

As a result, 6 different functions have been found for objective function values of PD-Clustering and fuzzy clustering problems. These functions are nothing but linear functions depending on n. If we plot these functions to see the minimum one, the plots given in Figure A.2 are obtained. As could be seen in the figure, in both problems, as n gets larger, objective function value of Case 2 becomes the minimum. To be more

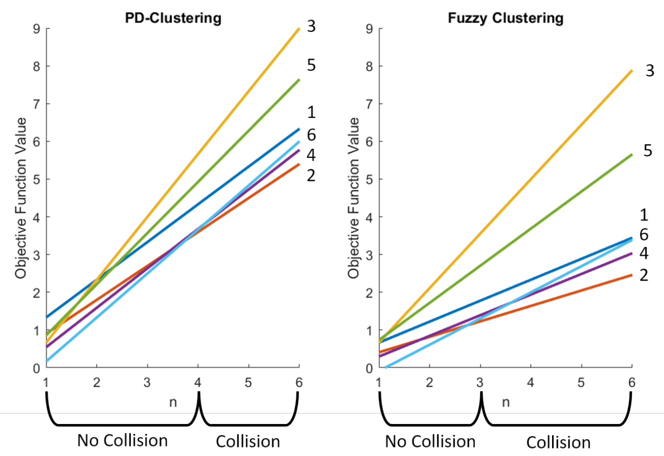

Figure A.2: Objective function change of each case depending on n

precise, Case 2 is minimum for n values greater than 4 for PD-Clustering, and for n values greater than 3 for Fuzzy Clustering with m=3. Case 2 is one of the cases that centers collide. Therefore, it is concluded that center collision is observed on the H-Tree for both problems.

Appendix B

PSEUDOCODES FOR DATA SIMULATION

B.1 Pseudocode for Uniform Graph Simulation

Algorithm 11 Uniform Graph Generation

Input: $numberOfVertices, NumberOfEdges$
Output: $Graph$
$verticesGenerated = 0, coordVertices = \emptyset$ ▷ Vertex coordinate generation
$scale = \sqrt{numberOfVertices}$
while $verticesGenerated <= numberOfVertices$ **do**
    Generate normal random (x, y) coordinates with $\mu = scale/2$ and $\sigma = scale/6$
    if verticesGenerated<1 **then**
        Add new (x, y) to $coordVertices$
        $verticesGenerated = verticesGenerated + 1$
    else
        Calculate Euclidean distance between new vertex and existing vertices
        if Distance to any vertex is less than $2 * scale/numberOfVertices$ **then**
            Continue
        else
            Add new (x, y) to $coordVertices$
            $verticesGenerated = verticesGenerated + 1$
        end if
    end if
end while

Algorithm 11 Uniform Graph Generation (continued)

$Edges = \emptyset$ ▷ Find Random Spanning Tree
$Connected = \emptyset$, $notConnected = V$ ▷ V is set of vertices
Select a starting vertex stV randomly
Find set of vertices $nearest$ whose Euclidean distance to stV is less than $(scale/\sqrt{numberOfVertices})$
if $nearest = \emptyset$ **then**
    Select the vertex that has the minimum Euclidean distance to stV and store it as enV
else
    Select a vertex $enV \in nearest$ randomly
end if
$edgesGenerated = 1$, $Connected = Connected \cup \{stV, enV\}$
$notConnected = notConnected - \{stV, enV\}$, $Edges = Edges \cup \{stV, enV\}$
while $notConnected \neq \emptyset$ **do**
    Find the vertex pairs $(Connected, notConnected)$ the Euclidean distance between which is less than $sqrt(2) * scale/sqrt(nVertices)$ and store them as $(nearestS, nearestE)$
    if $nearestS \neq \emptyset$ **then**
        Select a $newEdge = (stV, enV)$ randomly from $(nearestS, nearestE)$
    else
        Select the $newEdge = (stV, enV)$ with the minimum Euclidean distance
    end if
    $Edges = Edges \cup newEdge$
    $edgesGenerated = edgesGenerated + 1$, $Connected = Connected \cup \{stV, enV\}$
    $notConnected = notConnected - \{stV, enV\}$
end while
$Graph = (Edges, V, coordVertices)$

Algorithm 11 Uniform Graph Generation (continued)

Find degrees of each vertex and store these in $Degree$
Find shortest path distances on $Graph$
$edgeCount = 1$
while $edgeCount \leq NumberOfEdges - edgesGenerated$ **do**
    Find the set of vertices with maximum degree $maxDeg$
    $canConnect = V - maxDeg$ ▷ V is set of vertices
    if $edgeCount \leq (NumberOfEdges - edgesGenerated)/2$ **then**
        if $|canConnect| > 2$ **then** ▷ Find two end vertices stV, enV
            Select a random $stV \in canConnect$
            Select the vertex $enV \in canConnect$ closest to stV on plane
        else if $|canConnect| \geq 1$ **then**
            Select a random $stV \in canConnect$
            Select the vertex $enV \in maxDeg$ closest to stV on plane
        else
            Select a random $stV \in maxDeg$
            Select the vertex $enV \in maxDeg$ closest to stV on plane
        end if
    else
        if $|canConnect| > 2$ **then** ▷ $ratio$ is the ratio of length of the shortest path on the graph to Euclidean distance on the plane
            Select $stV \in canConnect$ and $enV \in canConnect$ with max $ratio$
        else if $|canConnect| \geq 1$ **then**
            Select $stV \in canConnect$ and $enV \in maxDeg$ with max $ratio$
        else
            Select $stV \in maxDeg$ and $enV \in maxDeg$ with max $ratio$
        end if
    end if

Algorithm 11 Uniform Graph Generation (continued)

  if $newEdge$ is duplicate OR $length(newEdge) \geq 0.75 * scale * \sqrt{2}$ **then**
    Do not add $newEdge$ and **Continue**
  else
    $Edge = Edge \cup newEdge$, update $Degree$
    Update distance between stV, enV as ∞
    $edgesGenerated = edgesGenerated + 1$
  end if
end while
$Graph = (Edges, V, coordVertices)$

B.2 Pseudocode for Random Graph Simulation

Algorithm 12 Random Graph Generation

Input: $numberOfVertices, NumberOfEdges$
Output: $Graph$
$verticesGenerated = 0, coordVertices = \emptyset$ ▷ Vertex coordinate generation
$scale = \sqrt{numberOfVertices}$
while $verticesGenerated <= numberOfVertices$ **do**
  Generate uniform random (x, y) coordinates in [0,scale]
  if verticesGenerated<1 **then**
    Add new (x, y) to $coordVertices$
    $verticesGenerated = verticesGenerated + 1$
  else
    Calculate distance between new vertex and existing vertices
    if Distance to any vertex is less than $2 * scale/numberOfVertices$ **then**
      Continue
    else

Algorithm 12 Random Graph Generation (continued)

      Add new (x, y) to $coordVertices$
      $verticesGenerated = verticesGenerated + 1$
    end if
  end if
end while
Find Minimum Spanning Tree by Prim's Algorithm [32] and store edges in $Edge(startVertex, endVertex)$
$edgesGenerated = numberOfVertices - 1$ ▷ Random edge generation
Find degrees of each vertex and store these in $Degree$
while $edgesGenerated \leq NumberOfEdges$ **do**
  Find the set of vertices with maximum degree $maxDeg$
  $canConnect = V - maxDeg$ ▷ V is set of vertices
  if $|canConnect| > 2$ **then** ▷ Find two end vertices stV, enV
    Select a random $stV \in canConnect$
    Select the vertex $enV \in canConnect$ closest to stV on plane
  else if $|canConnect| \geq 1$ **then**
    Select a random $stV \in canConnect$
    Select the vertex $enV \in maxDeg$ closest to stV on plane
  else
    Select a random $stV \in maxDeg$
    Select the vertex $enV \in maxDeg$ closest to stV on plane
  end if
  $newEdge = (stV, enV)$
  if $newEdge$ is duplicate OR $length(newEdge) \geq 0.75 * scale * \sqrt{2}$ **then**
    Do not add $newEdge$ and **Continue**
  else
    $Edge = Edge \cup newEdge$, update $Degree$
    Update distance between stV, enV as ∞
    $edgesGenerated = edgesGenerated + 1$
  end if
end while
$Graph = (Edges, V, coordVertices)$

Appendix C

COMPUTATIONAL RESULTS OF SIMULATED DATA

In this chapter, outputs of the computational studies conducted with simulated data sets have been provided. Two types of problems (PD-Clustering and Fuzzy Clustering) have been solved with HGA and Heuristics modified for instances, and all output is reported. In these computational experiments, 5 replications were performed for each instance with HGA algorithms, and 10 replications were performed for each instance with the modified heuristics.

Table C.1: HGA-N results for PD-Clustering Problem for *Uniform* instances

Instance	**Vertices-Clusters-Edges**	**Best Found Value**	**Avg % Dev**	**Worst % Dev**	**Runtime (sec)**
Unif1	10-3-20	213.26	0.72	1.49	0.03
Unif2	10-5-20	87.64	-	-	0.06
Unif3	30-2-60	2015.38	-	-	0.05
Unif4	30-5-60	731.51	-	-	0.14
Unif5	30-10-60	284.49	0.14	0.70	0.31
Unif6	50-2-100	4563.88	-	-	0.07
Unif7	50-5-100	1583.40	-	-	0.26
Unif8	50-10-100	749.22	0.02	0.05	0.66
Unif9	100-5-200	4833.39	0.01	0.03	0.43
Unif10	100-10-200	2332.16	-	-	1.16
Unif11	100-10-200	2329.87	-	-	1.24
Unif12	100-20-200	1005.43	0.01	0.03	3.78
Unif13	100-34-200	480.87	0.06	0.11	9.89
Unif14	200-5-400	13649.64	-	-	0.62
Unif15	200-10-400	6847.57	0.01	0.04	2.40
Unif16	200-20-400	3271.08	-	0.02	8.83
Unif17	200-40-400	1433.28	0.02	0.04	23.40
Unif18	200-67-400	698.51	0.03	0.07	64.96
Unif19	200-5-800	12681.06	0.01	0.03	0.65
Unif20	200-10-800	6060.36	-	0.02	2.37
Unif21	200-20-800	2913.24	0.02	0.04	8.85
Unif22	200-40-800	1290.65	0.02	0.06	26.19
Unif23	200-67-800	653.25	0.03	0.05	52.07
Unif24	300-5-600	25856.59	0.05	0.24	0.85
Unif25	300-10-600	12597.72	-	0.02	3.28
Unif26	300-30-600	3805.94	0.02	0.03	20.62
Unif27	300-60-600	1705.46	0.03	0.05	67.59
Unif28	300-100-600	857.14	0.03	0.09	177.67
Unif29	300-5-1200	23440.09	-	-	0.86
Unif30	300-10-1200	11066.61	-	-	3.20
Unif31	300-30-1200	3520.18	0.02	0.06	22.05
Unif32	300-60-1200	1550.89	0.02	0.04	74.26
Unif33	300-100-1200	801.13	0.02	0.04	142.57
Unif34	300-5-1800	22761.22	0.01	0.03	0.93
Unif35	300-10-1800	11020.74	0.03	0.09	3.67
Unif36	300-30-1800	3441.03	0.01	0.02	27.73
Unif37	300-60-1800	1562.45	0.01	0.03	89.06
Unif38	300-100-1800	793.18	0.01	0.04	171.27
Unif39	400-5-800	39692.01	0.01	0.03	1.19
Unif40	400-10-800	19528.20	-	-	4.47
Unif41	400-40-800	4475.42	0.04	0.07	46.78
Unif42	400-80-800	1964.99	0.01	0.03	142.04
Unif43	400-134-800	1002.30	0.03	0.06	344.83
Unif44	400-5-1600	35381.85	-	0.02	1.11
Unif45	400-10-1600	17452.06	0.01	0.02	4.29
Unif46	400-40-1600	4040.13	0.02	0.04	49.79
Unif47	400-80-1600	1809.82	0.03	0.05	158.22
Unif48	400-134-1600	904.75	0.01	0.02	342.23
Unif49	400-5-2400	35137.15	-	-	1.30
Unif50	400-10-2400	17124.52	0.01	0.03	4.80
Unif51	400-40-2400	3927.64	0.01	0.02	58.96
Unif52	400-80-2400	1770.82	0.01	0.03	172.13
Unif53	400-134-2400	889.26	0.01	0.03	432.86
Unif54	400-5-3200	34913.11	0.02	0.04	1.30
Unif55	400-10-3200	17183.09	0.01	0.06	5.52
Unif56	400-40-3200	3937.07	0.01	0.03	62.04
Unif57	400-80-3200	1776.13	0.02	0.03	201.35
Unif58	400-134-3200	887.91	0.02	0.04	436.90
Unif59	600-5-1200	71662.16	0.08	0.20	1.70
Unif60	600-10-2400	31852.59	0.01	0.03	7.43
	Average		**0.03**	**0.07**	**58.25**

Table C.2: M-*PD-Clustering* results for PD-Clustering Problem for *Uniform* instances

Instance	**Vertices-Clusters-Edges**	**Best Found Value**	**Avg % Dev**	**Worst % Dev**	**Runtime (sec)**
Unif1	10-3-20	213.26	4.17	7.66	2.17
Unif2	10-5-20	92.62	6.27	22.67	0.74
Unif3	30-2-60	2015.82	1.00	4.03	1.14
Unif4	30-5-60	745.46	3.50	12.68	0.91
Unif5	30-10-60	295.89	3.60	7.41	1.28
Unif6	50-2-100	4563.88	2.37	3.44	1.58
Unif7	50-5-100	1610.42	2.62	4.84	0.99
Unif8	50-10-100	783.18	2.11	5.37	0.79
Unif9	100-5-200	4921.43	1.03	2.22	0.95
Unif10	100-10-200	2429.60	1.43	3.44	0.89
Unif11	100-10-200	2425.52	2.05	5.09	0.90
Unif12	100-20-200	1050.63	2.26	4.87	1.61
Unif13	100-34-200	502.69	2.03	3.77	1.07
Unif14	200-5-400	13782.08	1.50	2.76	1.49
Unif15	200-10-400	6970.27	2.38	4.41	1.13
Unif16	200-20-400	3396.60	2.47	5.09	1.10
Unif17	200-40-400	1506.83	1.18	4.25	1.43
Unif18	200-67-400	724.41	0.92	2.67	1.56
Unif19	200-5-800	12714.64	0.44	1.08	1.64
Unif20	200-10-800	6159.83	0.98	1.64	1.17
Unif21	200-20-800	3020.56	1.64	5.86	1.92
Unif22	200-40-800	1326.52	2.39	4.31	1.31
Unif23	200-67-800	676.47	1.05	2.12	1.65
Unif24	300-5-600	26021.06	2.12	3.17	1.76
Unif25	300-10-600	12855.36	1.04	1.90	1.38
Unif26	300-30-600	3990.97	1.20	4.03	1.38
Unif27	300-60-600	1794.60	1.53	4.20	1.89
Unif28	300-100-600	890.62	0.87	2.27	2.07
Unif29	300-5-1200	23475.62	0.53	0.99	1.77
Unif30	300-10-1200	11213.98	0.74	1.46	2.21
Unif31	300-30-1200	3640.19	1.82	3.88	1.53
Unif32	300-60-1200	1604.06	1.14	2.61	1.75
Unif33	300-100-1200	826.07	0.61	1.22	2.18
Unif34	300-5-1800	22778.70	0.33	0.73	1.71
Unif35	300-10-1800	11144.15	0.85	2.22	2.17
Unif36	300-30-1800	3546.49	2.71	4.26	1.59
Unif37	300-60-1800	1616.21	1.12	2.50	1.90
Unif38	300-100-1800	820.02	1.08	1.64	2.43
Unif39	400-5-800	40212.43	0.33	0.71	2.45
Unif40	400-10-800	19850.44	1.36	3.04	2.58
Unif41	400-40-800	4690.10	1.73	5.50	1.93
Unif42	400-80-800	2045.70	1.12	2.61	2.33
Unif43	400-134-800	1030.18	1.29	2.15	2.77
Unif44	400-5-1600	35427.14	0.19	0.47	2.40
Unif45	400-10-1600	17685.34	0.38	1.20	1.97
Unif46	400-40-1600	4222.24	1.61	3.36	1.93
Unif47	400-80-1600	1877.56	1.03	1.68	2.29
Unif48	400-134-1600	930.77	1.19	2.27	3.58
Unif49	400-5-2400	35164.04	0.30	0.75	2.37
Unif50	400-10-2400	17281.96	0.52	1.31	2.02
Unif51	400-40-2400	4057.30	2.00	3.92	2.14
Unif52	400-80-2400	1843.66	1.33	3.59	2.50
Unif53	400-134-2400	909.78	1.40	2.53	3.19
Unif54	400-5-3200	34949.88	0.20	0.64	2.51
Unif55	400-10-3200	17332.06	0.32	1.20	2.12
Unif56	400-40-3200	4073.71	1.85	5.83	2.16
Unif57	400-80-3200	1837.27	1.70	3.11	2.58
Unif58	400-134-3200	914.74	0.81	2.09	3.20
Unif59	600-5-1200	72079.02	1.13	2.70	3.26
Unif60	600-10-2400	32020.83	0.45	0.86	2.49
	Average		**1.49**	**3.50**	**1.86**

Table C.3: HGA-N results for PD-Clustering Problem for *Random* instances

Instance	Vertices-Clusters-Edges	Best Found Value	Avg % Dev	Worst % Dev	Runtime (sec)
Rand1	10-3-20	171.79	-	-	0.04
Rand2	10-5-20	71.85	0.07	0.34	0.05
Rand3	30-2-60	1652.45	-	-	0.05
Rand4	30-5-60	551.51	0.06	0.30	0.16
Rand5	30-10-60	245.42	0.01	0.04	0.37
Rand6	50-2-100	3958.35	0.02	0.10	0.06
Rand7	50-5-100	1449.71	0.01	0.07	0.28
Rand8	50-10-100	592.15	-	-	0.66
Rand9	100-5-200	3917.06	-	-	0.36
Rand10	100-10-200	1789.41	-	0.01	1.09
Rand11	100-10-200	1860.13	-	-	1.18
Rand12	100-20-200	788.38	-	-	3.46
Rand13	100-34-200	393.33	-	-	12.41
Rand14	200-5-400	12054.54	-	-	0.61
Rand15	200-10-400	5879.24	0.01	0.01	1.98
Rand16	200-20-400	2645.00	0.01	0.02	7.56
Rand17	200-40-400	1146.90	-	-	26.04
Rand18	200-67-400	553.17	0.03	0.08	69.07
Rand19	200-5-800	10470.84	-	-	0.66
Rand20	200-10-800	4865.25	-	-	1.97
Rand21	200-20-800	2172.54	-	0.01	8.63
Rand22	200-40-800	1003.63	0.01	0.03	28.97
Rand23	200-67-800	508.63	-	-	77.85
Rand24	300-5-600	22797.94	0.03	0.13	0.82
Rand25	300-10-600	10927.29	-	-	2.92
Rand26	300-30-600	3253.58	-	0.01	21.58
Rand27	300-60-600	1400.81	-	0.01	85.72
Rand28	300-100-600	709.90	0.04	0.09	231.47
Rand29	300-5-1200	19002.21	-	-	0.91
Rand30	300-10-1200	9021.96	-	-	2.96
Rand31	300-30-1200	2772.83	-	0.01	20.96
Rand32	300-60-1200	1242.36	-	-	87.21
Rand33	300-100-1200	623.10	0.03	0.05	242.33
Rand34	300-5-1800	18587.80	-	-	0.88
Rand35	300-10-1800	8744.83	-	-	2.92
Rand36	300-30-1800	2657.48	-	-	22.67
Rand37	300-60-1800	1179.35	0.01	0.01	95.47
Rand38	300-100-1800	582.16	0.02	0.04	290.25
Rand39	400-5-800	35321.72	0.01	0.05	1.00
Rand40	400-10-800	16440.96	0.04	0.11	4.34
Rand41	400-40-800	3715.01	0.01	0.02	47.61
Rand42	400-80-800	1696.47	-	-	163.95
Rand43	400-134-800	844.87	0.06	0.10	518.92
Rand44	400-5-1600	29290.89	0.01	0.02	1.14
Rand45	400-10-1600	13914.49	-	-	3.75
Rand46	400-40-1600	3150.73	-	-	49.97
Rand47	400-80-1600	1401.28	-	-	187.00
Rand48	400-134-1600	708.22	0.03	0.05	551.02
Rand49	400-5-2400	29222.82	-	-	1.13
Rand50	400-10-2400	13828.19	-	-	4.91
Rand51	400-40-2400	3091.03	0.01	0.01	53.21
Rand52	400-80-2400	1364.23	-	0.01	209.39
Rand53	400-134-2400	689.15	0.02	0.03	662.51
Rand54	400-5-3200	28469.36	-	-	1.33
Rand55	400-10-3200	13481.58	-	-	4.62
Rand56	400-40-3200	3053.87	-	-	62.89
Rand57	400-80-3200	1367.72	-	0.01	221.90
Rand58	400-134-3200	669.21	0.01	0.02	788.17
Rand59	600-5-1200	64906.87	0.09	0.26	1.73
Rand60	600-10-2400	26152.89	0.01	0.05	7.36
	Average		**0.01**	**0.03**	**81.67**

Table C.4: M-*PD-Clustering* results for PD-Clustering Problem for *Random* instances

Instance	Vertices-Clusters-Edges	Best Found Value	Avg % Dev	Worst % Dev	Runtime (sec)
Rand1	10-3-20	171.79	7.06	33.80	0.86
Rand2	10-5-20	74.65	8.57	25.78	1.18
Rand3	30-2-60	1683.36	-	-	0.74
Rand4	30-5-60	556.34	7.39	17.72	0.73
Rand5	30-10-60	266.17	7.20	20.13	0.85
Rand6	50-2-100	3966.10	-	-	1.34
Rand7	50-5-100	1529.81	2.10	5.36	0.97
Rand8	50-10-100	637.65	5.95	11.40	1.60
Rand9	100-5-200	4008.95	3.91	5.77	1.42
Rand10	100-10-200	1896.52	4.41	10.51	0.98
Rand11	100-10-200	1967.49	2.31	4.99	0.97
Rand12	100-20-200	863.73	3.17	6.25	0.94
Rand13	100-34-200	424.05	3.45	5.62	1.00
Rand14	200-5-400	12441.54	1.54	3.23	2.01
Rand15	200-10-400	6287.66	1.61	3.35	1.37
Rand16	200-20-400	2809.87	4.68	8.77	1.10
Rand17	200-40-400	1231.55	2.97	5.92	1.22
Rand18	200-67-400	607.98	0.97	2.95	1.61
Rand19	200-5-800	10639.96	1.20	3.48	2.22
Rand20	200-10-800	4974.89	4.77	9.40	1.97
Rand21	200-20-800	2289.88	2.43	4.94	1.58
Rand22	200-40-800	1080.13	2.08	4.81	1.33
Rand23	200-67-800	549.46	1.63	3.55	2.03
Rand24	300-5-600	23244.76	2.33	4.51	2.00
Rand25	300-10-600	11467.04	2.16	5.71	2.09
Rand26	300-30-600	3531.87	3.26	5.37	1.47
Rand27	300-60-600	1520.18	1.27	3.31	2.16
Rand28	300-100-600	762.14	1.14	2.94	2.33
Rand29	300-5-1200	19055.84	0.87	2.00	2.16
Rand30	300-10-1200	9308.12	1.72	2.75	2.30
Rand31	300-30-1200	3004.32	1.54	3.76	1.55
Rand32	300-60-1200	1332.47	1.25	3.16	2.42
Rand33	300-100-1200	661.02	1.13	2.49	2.10
Rand34	300-5-1800	18659.37	1.24	2.99	2.39
Rand35	300-10-1800	9004.01	1.72	3.41	2.09
Rand36	300-30-1800	2840.51	2.00	3.40	1.65
Rand37	300-60-1800	1252.51	1.27	2.82	1.80
Rand38	300-100-1800	616.20	1.40	3.68	2.17
Rand39	400-5-800	35810.02	1.26	2.16	3.01
Rand40	400-10-800	17292.16	1.23	2.75	2.30
Rand41	400-40-800	3962.43	2.18	4.99	1.99
Rand42	400-80-800	1823.37	1.58	2.43	2.32
Rand43	400-134-800	901.66	2.19	3.99	2.84
Rand44	400-5-1600	29548.37	0.80	1.66	3.23
Rand45	400-10-1600	14349.68	1.47	2.89	2.43
Rand46	400-40-1600	3356.87	1.38	2.68	2.17
Rand47	400-80-1600	1497.81	0.65	2.85	2.32
Rand48	400-134-1600	752.78	0.85	1.93	2.94
Rand49	400-5-2400	29359.89	0.79	2.43	3.23
Rand50	400-10-2400	14080.58	1.54	2.70	2.58
Rand51	400-40-2400	3283.23	1.71	4.14	2.24
Rand52	400-80-2400	1453.68	1.06	3.30	3.32
Rand53	400-134-2400	727.97	0.61	1.80	2.99
Rand54	400-5-3200	28506.42	0.87	2.08	3.14
Rand55	400-10-3200	13842.32	1.96	4.92	3.26
Rand56	400-40-3200	3228.46	1.83	4.50	2.25
Rand57	400-80-3200	1453.16	1.38	4.00	2.73
Rand58	400-134-3200	697.31	1.22	2.17	3.25
Rand59	600-5-1200	66522.30	1.52	3.78	4.54
Rand60	600-10-2400	26638.87	0.94	2.05	4.27
	Average		**2.21**	**5.27**	**2.07**

Table C.5: HGA-E results for Fuzzy Clustering Problem for *Uniform* instances

Instance	Vertices-Clusters-Edges	Best Found Value	Avg % Dev	Worst % Dev	Runtime (sec)
Unif1	10-3-20	6655.58	0.88	2.72	1.24
Unif2	10-5-20	1556.71	0.01	0.02	2.21
Unif3	30-2-60	160131.33	-	-	1.16
Unif4	30-5-60	22152.84	-	-	5.78
Unif5	30-10-60	4125.11	1.19	2.35	15.53
Unif6	50-2-100	479214.10	-	-	1.62
Unif7	50-5-100	59611.87	0.07	0.36	6.89
Unif8	50-10-100	14451.55	0.33	0.59	20.57
Unif9	100-5-200	265868.13	0.40	1.23	11.72
Unif10	100-10-200	63812.88	0.16	0.65	37.03
Unif11	100-10-200	63827.80	0.34	0.74	35.67
Unif12	100-20-200	13116.98	0.25	0.39	123.26
Unif13	100-34-200	3623.30	0.54	0.79	297.78
Unif14	200-5-400	1047129.21	0.16	0.41	23.28
Unif15	200-10-400	262448.37	0.03	0.11	77.73
Unif16	200-20-400	62079.05	0.21	0.45	232.48
Unif17	200-40-400	13425.24	0.20	0.40	719.78
Unif18	200-67-400	3818.47	0.32	0.61	1695.89
Unif19	200-5-800	892697.57	0.07	0.18	64.03
Unif20	200-10-800	205094.98	0.23	0.51	244.18
Unif21	200-20-800	50154.47	0.04	0.08	895.96
Unif22	200-40-800	10919.94	0.13	0.29	3021.46
Unif23	200-67-800	3318.67	0.14	0.20	8135.81
Unif24	300-5-600	2474648.85	0.07	0.20	33.74
Unif25	300-10-600	586791.56	0.07	0.17	124.30
Unif26	300-30-600	56589.18	0.21	0.57	614.87
Unif27	300-60-600	12720.11	0.16	0.28	2168.65
Unif28	300-100-600	3842.52	0.28	0.45	6035.54
Unif29	300-5-1200	2039803.05	0.02	0.03	109.03
Unif30	300-10-1200	453307.97	0.13	0.29	393.55
Unif31	300-30-1200	48435.21	0.07	0.14	2692.02
Unif32	300-60-1200	10432.58	0.05	0.19	9738.68
Unif33	300-100-1200	3314.90	0.13	0.26	25134.16
Unif34	300-5-1800	1917230.46	0.32	0.87	192.72
Unif35	300-10-1800	448921.85	0.17	0.34	836.54
Unif36	300-30-1800	46281.74	0.03	0.10	5528.38
Unif37	300-60-1800	10577.11	0.06	0.19	19125.57
Unif38	300-100-1800	3252.36	0.09	0.16	52921.46
Unif39	400-5-800	4324419.85	0.70	1.02	42.30
Unif40	400-10-800	1057090.18	0.15	0.36	153.90
Unif41	400-40-800	58531.25	0.12	0.24	1405.21
Unif42	400-80-800	12675.22	0.16	0.36	4687.63
Unif43	400-134-800	3928.98	0.06	0.13	13600.78
Unif44	400-5-1600	3475104.54	0.09	0.24	144.65
Unif45	400-10-1600	838938.46	0.15	0.38	595.97
Unif46	400-40-1600	48077.07	0.05	0.10	6141.21
Unif47	400-80-1600	10699.23	0.11	0.26	20966.77
Unif48	400-134-1600	3178.91	0.11	0.20	58307.18
Unif49	400-5-2400	3417234.38	0.15	0.56	269.76
Unif50	400-10-2400	806814.99	0.12	0.25	1296.82
Unif51	400-40-2400	45121.41	0.06	0.16	12575.83
Unif52	400-80-2400	10233.99	0.06	0.14	44332.12
Unif53	400-134-2400	3065.18	0.09	0.17	132329.73
Unif54	400-5-3200	3358594.45	0.14	0.33	453.56
Unif55	400-10-3200	812646.65	0.11	0.32	1902.82
Unif56	400-40-3200	45363.98	0.07	0.17	20757.00
Unif57	400-80-3200	10347.27	0.03	0.06	72079.54
Unif58	400-134-3200	3054.60	0.07	0.12	213070.67
Unif59	600-5-1200	9476422.09	0.42	0.75	68.11
Unif60	600-10-2400	1853226.67	0.12	0.22	1033.24
	Average		**0.18**	**0.40**	**12458.92**

Table C.6: M-*Fuzzy C-Means* results for Fuzzy Clustering Problem for *Uniform* instances

Instance	Vertices-Clusters-Edges	Best Found Value	Avg % Dev	Worst % Dev	Runtime (sec)
Unif1	10-3-20	8859.17	2.75	3.92	1.05
Unif2	10-5-20	1617.13	7.44	18.26	0.86
Unif3	30-2-60	167675.92	-	-	1.80
Unif4	30-5-60	24222.26	2.35	4.34	1.51
Unif5	30-10-60	4533.42	3.96	8.91	1.11
Unif6	50-2-100	527752.67	-	-	2.27
Unif7	50-5-100	62723.42	0.24	2.38	2.85
Unif8	50-10-100	14978.02	4.67	9.57	1.45
Unif9	100-5-200	287149.10	0.45	3.32	3.09
Unif10	100-10-200	66531.01	2.45	5.15	2.41
Unif11	100-10-200	66860.66	1.68	5.50	2.02
Unif12	100-20-200	13536.41	6.75	9.51	1.82
Unif13	100-34-200	3889.82	3.71	8.48	2.25
Unif14	200-5-400	1151522.07	0.01	0.01	6.75
Unif15	200-10-400	272463.37	2.15	4.72	5.69
Unif16	200-20-400	65507.49	2.13	8.20	2.92
Unif17	200-40-400	14286.28	2.65	6.31	4.01
Unif18	200-67-400	4018.41	2.89	5.68	5.46
Unif19	200-5-800	1184062.22	-	-	4.81
Unif20	200-10-800	212317.08	6.39	18.46	5.15
Unif21	200-20-800	52140.66	2.32	10.10	4.07
Unif22	200-40-800	11432.29	4.00	8.58	5.85
Unif23	200-67-800	3453.65	3.71	10.33	8.69
Unif24	300-5-600	2647378.07	-	0.01	7.88
Unif25	300-10-600	610258.39	0.69	2.83	7.94
Unif26	300-30-600	59948.24	2.13	4.06	4.89
Unif27	300-60-600	13567.98	1.81	4.79	7.08
Unif28	300-100-600	4096.60	1.36	3.10	10.55
Unif29	300-5-1200	2160966.72	0.85	2.11	18.04
Unif30	300-10-1200	468558.73	7.83	30.27	9.19
Unif31	300-30-1200	50623.83	3.06	8.55	7.03
Unif32	300-60-1200	11056.87	2.10	4.88	11.45
Unif33	300-100-1200	3471.09	2.48	4.58	18.05
Unif34	300-5-1800	2626982.41	0.01	0.01	8.53
Unif35	300-10-1800	466590.34	6.14	14.13	11.79
Unif36	300-30-1800	49218.22	2.00	3.87	9.94
Unif37	300-60-1800	11123.44	2.11	3.85	16.14
Unif38	300-100-1800	3480.85	1.01	3.25	25.18
Unif39	400-5-800	4556817.00	0.89	3.98	8.37
Unif40	400-10-800	1096703.44	4.65	20.77	8.77
Unif41	400-40-800	62438.41	1.92	3.07	7.46
Unif42	400-80-800	13586.69	1.97	5.86	11.70
Unif43	400-134-800	4119.31	1.60	3.91	17.78
Unif44	400-5-1600	3724559.20	-	0.01	13.12
Unif45	400-10-1600	867975.34	7.06	15.82	11.43
Unif46	400-40-1600	50621.80	3.13	5.97	11.49
Unif47	400-80-1600	11243.61	2.50	4.26	19.23
Unif48	400-134-1600	3347.03	1.66	4.50	30.43
Unif49	400-5-2400	4119848.02	-	-	11.23
Unif50	400-10-2400	831725.77	11.97	28.21	13.20
Unif51	400-40-2400	47428.94	3.72	8.04	15.64
Unif52	400-80-2400	10843.78	2.05	4.87	26.92
Unif53	400-134-2400	3231.12	1.62	5.45	42.98
Unif54	400-5-3200	3889626.56	0.01	0.01	10.35
Unif55	400-10-3200	850686.80	11.78	23.85	14.00
Unif56	400-40-3200	47511.76	4.32	6.41	19.46
Unif57	400-80-3200	11039.63	1.78	4.68	34.65
Unif58	400-134-3200	3222.96	1.25	2.16	56.58
Unif59	600-5-1200	9882690.88	0.25	0.42	23.20
Unif60	600-10-2400	1931256.14	0.72	2.52	20.29
	Average		**2.69**	**6.61**	**11.33**

Table C.7: HGA-E Results for Fuzzy Clustering Problem for *Random* instances

Instance	Vertices-Clusters-Edges	Best Found Value	Avg % Dev	Worst % Dev	Runtime (sec)
Rand1	10-3-20	4322.92	-	-	0.81
Rand2	10-5-20	1047.41	0.01	0.03	1.65
Rand3	30-2-60	114353.98	-	-	1.14
Rand4	30-5-60	13461.07	0.05	0.14	4.84
Rand5	30-10-60	3095.19	0.04	0.09	16.66
Rand6	50-2-100	381819.64	0.26	0.84	1.51
Rand7	50-5-100	49577.60	-	-	7.13
Rand8	50-10-100	9118.79	0.07	0.14	22.21
Rand9	100-5-200	184413.36	-	-	9.67
Rand10	100-10-200	38521.14	-	0.01	34.58
Rand11	100-10-200	41668.27	0.01	0.04	34.34
Rand12	100-20-200	8154.10	0.06	0.16	120.08
Rand13	100-34-200	2407.44	0.60	1.27	323.36
Rand14	200-5-400	832510.89	0.31	1.12	17.35
Rand15	200-10-400	196558.92	-	-	58.09
Rand16	200-20-400	42076.19	0.11	0.27	178.97
Rand17	200-40-400	8677.84	0.12	0.27	759.00
Rand18	200-67-400	2421.18	0.23	0.49	1759.47
Rand19	200-5-800	619974.72	0.01	0.05	47.73
Rand20	200-10-800	137142.01	-	-	168.38
Rand21	200-20-800	28233.05	0.04	0.06	715.10
Rand22	200-40-800	6582.09	0.14	0.29	2403.86
Rand23	200-67-800	2004.43	0.19	0.39	6167.42
Rand24	300-5-600	1950729.78	0.07	0.17	26.13
Rand25	300-10-600	442659.30	0.15	0.25	95.96
Rand26	300-30-600	41926.55	0.29	0.50	581.83
Rand27	300-60-600	8574.09	0.24	0.57	1927.61
Rand28	300-100-600	2658.14	0.24	0.70	5060.00
Rand29	300-5-1200	1360727.48	-	-	76.33
Rand30	300-10-1200	306503.85	0.01	0.02	273.04
Rand31	300-30-1200	30125.46	0.09	0.14	1818.63
Rand32	300-60-1200	6738.31	0.23	0.48	7005.78
Rand33	300-100-1200	2003.18	0.34	0.71	18828.63
Rand34	300-5-1800	1288078.96	-	0.01	131.75
Rand35	300-10-1800	289619.10	0.02	0.09	530.42
Rand36	300-30-1800	27731.79	0.03	0.05	3583.54
Rand37	300-60-1800	6034.42	0.05	0.13	13862.36
Rand38	300-100-1800	1748.24	0.09	0.15	39138.58
Rand39	400-5-800	3463718.14	-	-	33.62
Rand40	400-10-800	760198.32	0.05	0.10	108.11
Rand41	400-40-800	40799.08	0.13	0.39	1181.04
Rand42	400-80-800	9434.29	0.10	0.18	3998.40
Rand43	400-134-800	2801.27	0.30	0.69	11214.83
Rand44	400-5-1600	2383757.76	0.05	0.16	83.24
Rand45	400-10-1600	548209.44	0.08	0.10	377.79
Rand46	400-40-1600	29387.36	0.14	0.26	4257.15
Rand47	400-80-1600	6381.86	0.17	0.33	14259.10
Rand48	400-134-1600	1949.63	0.16	0.48	42790.46
Rand49	400-5-2400	2375468.90	0.10	0.29	172.30
Rand50	400-10-2400	530390.82	0.03	0.16	696.92
Rand51	400-40-2400	28059.38	0.08	0.22	8072.04
Rand52	400-80-2400	6049.09	0.07	0.22	28975.33
Rand53	400-134-2400	1837.80	0.26	0.39	88997.17
Rand54	400-5-3200	2232532.18	0.04	0.20	274.18
Rand55	400-10-3200	513240.35	0.04	0.20	1350.56
Rand56	400-40-3200	27280.05	0.09	0.19	13002.71
Rand57	400-80-3200	6053.57	0.12	0.26	49737.58
Rand58	400-134-3200	1730.19	0.16	0.33	154880.69
Rand59	600-5-1200	7642367.47	0.07	0.33	62.36
Rand60	600-10-2400	1260072.74	0.24	0.41	706.25
	Average		**0.11**	**0.26**	**8850.43**

Table C.8: M-*Fuzzy C-Means* Results for Fuzzy Clustering Problem for *Random* instances

Instance	Vertices-Clusters-Edges	Best Found Value	Avg % Dev	Worst % Dev	Runtime (sec)
Rand1	10-3-20	4883.51		-	1.03
Rand2	10-5-20	1090.97	12.10	27.51	0.90
Rand3	30-2-60	133065.31	0.01	0.01	1.49
Rand4	30-5-60	14554.77	4.14	13.78	1.44
Rand5	30-10-60	3246.72	15.64	49.01	1.21
Rand6	50-2-100	401552.97	-	-	1.36
Rand7	50-5-100	56225.44	7.35	10.10	1.80
Rand8	50-10-100	10229.48	8.98	23.10	1.87
Rand9	100-5-200	210621.58	-	-	2.36
Rand10	100-10-200	41300.07	5.36	17.37	2.26
Rand11	100-10-200	45885.99	3.50	7.58	2.80
Rand12	100-20-200	9289.36	9.32	20.67	2.26
Rand13	100-34-200	2847.95	6.54	12.45	2.39
Rand14	200-5-400	874897.08	-	-	2.64
Rand15	200-10-400	215569.60	4.53	10.48	3.18
Rand16	200-20-400	46605.49	3.23	9.54	3.71
Rand17	200-40-400	10260.89	2.27	4.55	5.21
Rand18	200-67-400	2837.36	4.75	12.67	5.59
Rand19	200-5-800	681907.89	2.44	4.88	3.23
Rand20	200-10-800	148680.84	2.12	6.77	4.01
Rand21	200-20-800	31414.08	8.06	18.70	4.87
Rand22	200-40-800	7302.13	7.56	14.96	6.98
Rand23	200-67-800	2296.36	2.79	11.21	8.78
Rand24	300-5-600	2090362.18	2.92	5.84	8.57
Rand25	300-10-600	476489.76	2.80	7.11	6.47
Rand26	300-30-600	47529.67	3.76	8.71	6.87
Rand27	300-60-600	9700.77	3.82	7.03	7.79
Rand28	300-100-600	2939.21	2.76	5.04	10.96
Rand29	300-5-1200	1509220.88	7.64	14.10	4.93
Rand30	300-10-1200	321648.47	5.96	21.61	6.65
Rand31	300-30-1200	33756.87	4.82	8.41	9.40
Rand32	300-60-1200	7531.33	3.01	4.96	12.78
Rand33	300-100-1200	2236.38	2.70	6.42	17.84
Rand34	300-5-1800	1419247.99	0.20	0.68	4.98
Rand35	300-10-1800	322997.93	1.77	5.11	8.54
Rand36	300-30-1800	31735.50	2.59	6.78	10.95
Rand37	300-60-1800	6813.22	2.66	5.70	16.18
Rand38	300-100-1800	1963.89	1.95	5.87	24.89
Rand39	400-5-800	4541066.99	-		5.32
Rand40	400-10-800	827039.69	3.40	5.13	8.23
Rand41	400-40-800	45811.72	4.33	9.76	10.27
Rand42	400-80-800	10996.01	2.26	5.28	12.56
Rand43	400-134-800	3141.39	2.51	4.58	18.35
Rand44	400-5-1600	2601985.64		-	5.38
Rand45	400-10-1600	593105.39	5.23	9.88	6.89
Rand46	400-40-1600	33513.46	1.35	3.33	13.61
Rand47	400-80-1600	7171.00	2.25	5.14	19.86
Rand48	400-134-1600	2175.27	2.25	9.54	30.21
Rand49	400-5-2400	2679823.06	6.71	25.30	6.19
Rand50	400-10-2400	568011.53	4.10	8.60	8.28
Rand51	400-40-2400	32104.86	2.85	10.59	16.97
Rand52	400-80-2400	6726.87	4.79	9.79	27.46
Rand53	400-134-2400	2038.77	1.60	4.28	42.71
Rand54	400-5-3200	3285406.89	-	-	7.34
Rand55	400-10-3200	547134.08	4.08	9.12	9.37
Rand56	400-40-3200	30788.64	3.13	6.26	22.63
Rand57	400-80-3200	6867.96	2.67	5.35	35.64
Rand58	400-134-3200	1906.36	1.88	7.23	55.71
Rand59	600-5-1200	8772882.86	0.21	0.34	6.73
Rand60	600-10-2400	1339688.91	1.31	2.57	12.45
	Average		**3.61**	**8.85**	**10.19**