ขั้นตอนวิธีการเกาะกลุ่มข้อมูลแบบขั้วสุดขีดครึ่งวงโคจร

นางสาวเบญจพรรณ กวีเลิศพจนา

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาคณิตศาสตร์ประยุกต์และวิทยาการคณนา
ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์
คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2556

HALF-ORBITAL EXTREME POLE CLUSTERING ALGORITHM

Miss Benjapun Kaveelerdpotjana

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Science Program in Applied Mathematics and
Computational Science
Department of Mathematics and Computer Science
Faculty of Science
Chulalongkorn University
Academic Year 2013

Thesis Title	HALF-ORBITAL EXTREME POLE CLUSTERING ALGORITHM
By	Miss Benjapun Kaveelerdpotjana
Field of Study	Applied Mathematics and Computational Science
Thesis Advisor	Boonyarit Intiyot, Ph.D.
Thesis Co-advisor	Assistant Professor Krung Sinapiromsaran, Ph.D.

เบญจพรรณ กวีเลิศพจนา : ขั้นตอนวิธีการเกาะกลุ่มข้อมูลแบบขั้วสุดขีดครึ่งวงโคจร. (HALF-ORBITAL EXTREME POLE CLUSTERING ALGORITHM) อ.ที่ปรึกษาวิทยานิพนธ์หลัก: ดร.บุญฤทธิ์ อินทิยศ, อ.ที่ปรึกษาวิทยานิพนธ์ร่วม: ผศ. ดร.กรุง สินอภิรมณ์สราญ, 54 หน้า.

วิทยานิพนธ์นี้นำเสนอขั้นตอนวิธีการเกาะกลุ่มข้อมูลแบบขั้วสุดขีดครึ่งวงโคจร ขั้นตอนวิธีการจัดกลุ่มข้อมูลแบบใหม่ที่ถูกนำเสนอได้ประยุกต์แนวคิดของการเลือกขั้วสุดขีดในการแบ่งข้อมูลออกเป็นกลุ่ม ความแปรปรวนรวมถูกใช้เป็นเครื่องมือในการวัดประสิทธิภาพเพื่อใช้เปรียบเทียบขั้นตอนวิธีที่เสนอกับขั้นตอนวิธีการจัดกลุ่มแบบ k-means และ k-medoids ขั้นตอนวิธีการเกาะกลุ่มข้อมูลแบบขั้วสุดขีดครึ่งวงโคจรได้แก้ไขจุดด้อยสามจุดของขั้นตอนวิธีที่ถูกนำมาเปรียบเทียบ คือ การไม่จำเป็นต้องระบุจำนวนของกลุ่มข้อมูลก่อนเริ่มขั้นตอนวิธี การใช้ระยะห่างระหว่างข้อมูลโดยไม่ต้องคำนวณซ้ำ และการให้ผลลัพธ์ของการจัดกลุ่มเหมือนกันทุกครั้งสำหรับชุดข้อมูลเดียว

ภาควิชา	คณิตศาสตร์และวิทยาการคอมพิวเตอร์	ลายมือชื่อนิสิต ....
		ลายมือชื่อ อ.ที่ปรึกษาวิทยานิพนธ์หลัก ....
สาขาวิชา	คณิตศาสตร์ประยุกต์และวิทยาการคณนา	ลายมือชื่อ อ.ที่ปรึกษาวิทยานิพนธ์ร่วม ....
ปีการศึกษา	2556	

5472015023 : MAJOR APPLIED MATHEMATICS AND COMPUTATIONAL SCIENCE
KEYWORDS: EXTREME POLE / VECTOR CORE / CLUSTERING ALGORITHM

BENJAPUN KAVEELERDPOTJANA: HALF-ORBITAL EXTREME POLE CLUSTERING ALGORITHM. ADVISOR: BOONYARIT INTIYOT, Ph.D., CO-ADVISOR: ASST. PROF. KRUNG SINAPIROMSARAN, Ph.D., 54 pp.

This thesis proposes the half-orbital extreme pole clustering algorithm. The new proposed clustering algorithm applies the idea of the farthest pair to partition instances into groups. The total variance is used as the performance measure to compare proposed algorithm with k-means and k-medoids clustering algorithms. Half-orbital extreme pole clustering algorithm can rectify the three drawbacks of compared algorithms which are ignoring the number of clusters, using the original distances in every iteration and producing the same final clusters for a given dataset.

Department:	Mathematics and Computer Science	Student's Signature ....
		Advisor's Signature ....
Field of Study:	Applied Mathematics and Computational Science	Co-advisor's Signature ....
Academic Year:	2013	

ACKNOWLEDGEMENTS

I would like to express my sincere thanks to my advisor Dr. Boonyarit Intiyot and my co-advisor Assistant Professor Dr. Krung Sinapiromsaran for many helpful discussions and suggestions. When I faced with many obstacles in my life, they always gave me a lot of encouragement throughout the Master degree program, not only the research methodologies but also many other methodologies in life. I could not complete this thesis without their support.

Next, I would like to thank Dr. Petarpa Boonserm, Dr. Arthorn Luangsodsai and Associate Professor Dr. Montri Maleewong my thesis committees for their comments and suggestions.

Moreover, I wish to thank Applied Mathematics and Computational Science Program in the Department of Mathematics and Computer Science, Faculty of Science, Chulalongkorn University and The Development and Promotion of Science and Technology Talents project (DPST) for financial and technical support. They provided me several precious moments of my life and gave me a chance to educate in the most reputable university in Thailand.

Furthermore, I am thankful to my family and my friends especially Wacharasak Siriseriwam, Suebkul Kanchanasuk, Panote Songwattanasiri, Chareonchai Sirisomboonrat and Nattorn Buthong for all their support throughout the period of this research.

CONTENTS

Page

THAI ABSTRACT iv

ENGLISH ABSTRACT v

ACKNOWLEDGEMENTS vi

CONTENTS vii

LIST OF TABLES ix

LIST OF FIGURES x

CHAPTER I INTRODUCTION 1

1.1 Motivation and literature surveys 1

1.2 Research objective 3

1.3 Thesis overview 3

CHAPTER II BACKGROUND KNOWLEDGE 4

2.1 Metric 4

2.2 Clustering algorithm 5

2.2.1 K-means algorithm 5

2.2.2 K-medoids algorithm 8

2.3 Performance measure 11

2.3.1 Total Variances 11

2.4 Pre-processing 13

2.5 Literature review 13

2.5.1 Multi-attributed frame 13

2.5.2 Multi-attributed lens 14

2.6 Background concept 15

2.6.1 The farthest pair in clustering 16

2.6.2 Dispersion 16

2.6.2.1 Sturge’s formula 17
CHAPTER III HALF-ORBITAL EXTREME POLE CLUSTERING ALGORITHM 19
3.1 The notations for Half-Orbital Extreme Pole algorithm 19
3.2 Half-Orbital Extreme Pole algorithm 20
3.3 The pseudo code of HOEP algorithm 23
CHAPTER IV EXPERIMENTS AND RESULTS 26
4.1 Dataset description 26
4.1.1 Iris dataset 26
4.1.2 Seed dataset 27
4.1.3 E.coli dataset 28
4.1.4 Wine dataset 29
4.2 Results of the experiments 30
CHAPTER V CONCLUSION 39
REFERENCES 41
APPENDIX 44
APPENDIX : RESULTS FROM HALF-ORBITAL ALGORITHM 45
VITA 54

LIST OF TABLES

Page

Table 1 The statistical information of iris dataset 26
Table 2 The statistical information of seed dataset 27
Table 3 The statistical information of E.col dataset 28
Table 4 The statistical information of wine dataset........ 29
Table 5 Dataset Description 30
Table 6 Results of iris dataset from HOEP algorithm 31
Table 7 Comparison of the total variance of iris dataset 32
Table 8 Results of seed dataset from HOEP algorithm 33
Table 9 Comparison of the total variance of seed dataset........ 34
Table 10 Results of E.coli dataset from HOEP algorithm 35
Table 11 Comparison of the total variance of E.coli dataset 36
Table 12 Results of wine dataset from HOEP algorithm 37
Table 13 The comparison of the total variance of wine dataset........ 38
Table 14 The results from iris dataset 45
Table 15 The results from seed dataset 48
Table 16 The results from E.coli dataset 51
Table 17 The results from wine dataset........ 52

LIST OF FIGURES

Page

Figure 1 Three regions are partitioned into a frame by a vector core. 14
Figure 2 All instances having the same target class with the extreme poles lie in the lens. 15
Figure 3 Positively skewed curve, normal curve and negatively skewed curve respectively... 17
Figure 4 The examples of dispersion in nonhomogeneous.. 17
Figure 5 The vector core is divided into n intervals of equal width α. 21
Figure 6 p_1 is chosen as the center of balls with several determined radiuses. 21
Figure 7 The desirable layer containing a splitting point is specified by parameter gamma. 22
Figure 8 The ball $B(p_1; \frac{(2j-1)}{2}\alpha)$ is created and all instances are divided into two groups which are instances inside and outside the ball. 22
Figure 9 All instances in the ball lie on a half of ball with side along vector core. 23
Figure 10 The flowchart of HOEP algorithm.. 25
Figure 11 Iris dataset 26
Figure 12 Seed dataset... 27
Figure 13 E.coli dataset 28
Figure 14 Wine dataset.. 29
Figure 15 Result of iris dataset from HOEP algorithm with the first transformed function and the gamma value as 0.25 46
Figure 16 Result of iris dataset from HOEP algorithm with the second transformed function and the gamma value as 0.35 46
Figure 17 Result of iris dataset from k-means algorithm (k = 3)..................................... 47
Figure 18 Result of iris dataset from k-medoids algorithm (k = 3).................................. 47
Figure 19 Result of seed dataset from HOEP algorithm with the first transformed function and the gamma value as 0.10 49
Figure 20 Result of seed dataset from k-means algorithm (k = 4).................................. 49
Figure 21 Result of seed dataset from k-medoids algorithm (k = 4)................................ 49

Figure 22 Result of seed dataset from HOEP algorithm with the second transformed function and the gamma value as 0.1 50
Figure 23 Result of seed dataset from k-means algorithm (k = 2)............ 50
Figure 24 Result of seed dataset from k-medoids algorithm (k = 2)............ 50
Figure 25 Results of E.coli dataset from three algorithms are the same............ 53
Figure 26 Results of wine dataset from three algorithms are the same............ 53

CHAPTER I
INTRODUCTION

1.1 Motivation and literature surveys

Nowadays, the technology of data warehouse is rapidly developed. There are billions of information available to be archived. As the result of numerous data, data mining can help us to discover information or knowledge which is hidden in a database. Data mining, which has known in terms of the knowledge discovery in database (KDD), is the process of discovering hidden information or hidden knowledge from large databases. Data mining can be applied in many areas of research, such as market basket analysis in business; biomedical, bioinformatics, and genetics in area of sciences; and wireless sensor network in engineering.

Data mining can be separated into two major types which are supervised learning and unsupervised learning. The supervised learning concerns with a dataset whose target class of instances is given. This type of learning consists of two processes which are the training step and the predicting step. In the training step, the training dataset is used to construct the model to capture characteristics or patterns that lead to a given target class. Then, the constructed model is used to predict the target class of unknown instances in predicting step. In other word, the supervised learning tries to construct model from known instances in order to classify unknown instances. There are several techniques in the supervised learning, such as decision tree [1], k-nearest neighbors [2], naive Bayesian algorithm [3], and support vector machine (SVM) [4]. On the other hand, the unsupervised learning does not need the target class. It finds the hidden structure in the unlabeled instances. There are several approaches in the unsupervised learning, such as principal component analysis (PCA) [5], self-organizing map (SOM) [6], and clustering algorithm. In this thesis, we propose a clustering algorithm.

A clustering algorithm is a procedure to organize instances into groups (clusters), such that instances in the same group are more similar to one another than instances in other groups. There are several techniques in clustering such as k-means [7, 8, 9], PAM [10], CLARA [11] and CLARANS [12]. However, k-means algorithm is one of the widely used techniques because of its efficiency and simplicity.

K-means algorithm was first proposed by Lloyd in 1957 but it was published in 1982 [9]. However, MacQueen is the first one who used the term “k-means” in 1967. In the beginning, MacQueen proposed k-means algorithm [7] that each cluster is represented by the center of the cluster (centroid). There are several weaknesses of this algorithm. For example, it is sensitive to outliers. So the existence of outliers could strongly affect the result of clustering. Another drawback is that the number of clusters is required to be determined in advance. In addition, the algorithm is not suitable for discovering clusters with non-convex shapes. Moreover, computing the centroid is not reasonable for categorical variables. Later, k-medoids or Partition around medoids (PAM) introduced by Kaufman and Rousseeuw in 1987 [10] was proposed to fix the problem in the aspect of interpretation categorical data by changing the representative of a cluster from centroid to be one of the instances in the cluster called medoid. Furthermore, PAM is more robust than k-means with respect to outliers because outliers or other extreme values have less influenced on a medoid than a centroid.

However, PAM is not effective for large datasets. Hence, in 1990, Kaufman and Rousseeuw suggested the Clustering for LARge Applications algorithm (CLARA) [11] for dealing with the large dataset. Instead of finding medoids for the entire dataset, CLARA draws samples from the dataset, and then applies the PAM algorithm on each sample and gives the best clustering as the output. Even though CLARA can deal with the large dataset, one weakness of CLARA is that the efficiency of the algorithm depends on the sample size. Another weakness is that a good clustering based on samples might not lead to a good clustering of the whole dataset if the samples are biased.

Admittedly, k-means and k-medoids clustering are very simple and reasonably fast algorithms [13]. However, they have three major disadvantages. First, the number of clusters must be predetermined. Second, the distances between every instance and centroids have to be recalculated in every iteration. Third, the different initial centroids can result in different final clusters.

In this thesis, we propose a new clustering algorithm called Half-Orbital Extreme Pole (HOEP), which is proposed to rectify these drawbacks. HOEP algorithm applies the idea of the farthest pair from “Network intrusion detection by using multi-attributed frame decision tree” [14] and “Breast Cancer Diagnosis using Multi-Attributed Lens Recursive Partitioning Algorithm” [15]. Even though both papers dealt with decision tree which is a technique for classification, but the concept of the farthest pair is also applicable for clustering.

1.2 Research objective

The goal of this research is to obtain a new clustering algorithm called Half-Orbital Extreme Pole (HOEP). The proposed algorithm is implemented and its performance (total variance) is compared with ones from k-means and k-medoids clustering algorithms.

1.3 Thesis overview

The rest of the thesis is organized as follows.

In Chapter II, we present the background knowledge which includes k-means algorithm, k-medoids algorithm, performance measure, literature reviews and background concepts. Then, the half-orbital extreme pole clustering algorithm is presented in Chapter III. In Chapter IV, the experiments and results are shown. In Chapter V, we discuss the results and draw conclusions. Some future research ideas are also suggested in this chapter.

CHAPTER II
BACKGROUND KNOWLEDGE

In this chapter, we discuss the background knowledge that is important to this thesis. We divide this chapter into six main parts. First, the definition of metric is defined. Second, we introduce the concept of clustering algorithm and algorithms which are used to compare with our algorithm. Third, we introduce the total variance which is the performance measure used in this thesis. Fourth, we show the formula that is used in data pre-processing. Fifth, we review the literature that inspires the idea of this work. Finally, we state the background concepts of our algorithm including the farthest pair in clustering and dispersion.

2.1 Metric

Definition Let D be an arbitrary set. A function $\hat{d}: D \times D \rightarrow \mathbb{R} \cup \{\infty\}$ is a *metric* on D if the following conditions are satisfied for all $x, y, z \in D$.

1. Positiveness: $\hat{d}(x, y) > 0$ if $x \neq y$, and $\hat{d}(x, x) = 0$.
2. Symmetry: $\hat{d}(x, y) = \hat{d}(y, x)$.
3. Triangle inequality: $\hat{d}(x, z) \leq \hat{d}(x, y) + \hat{d}(y, z)$.

A metric space is a set with a metric on it. In other words, a metric space is a pair $(D, \hat{d})$ where $\hat{d}$ is a metric on D. Elements of D are called instances. $\hat{d}(x, y)$ is referred to the distance between instances x and y.

Let x and y be the instances in D which is a set of the d-dimensional real-value vectors.

1. Manhattan distance is defined by

$$\widehat{d_1}(x, y) = \sum_{i=1}^{d} |x_i - y_i| \tag{2.1}$$

2. Euclidean distance is defined by

$$\widehat{d_2}(x, y) = \sqrt{\sum_{i=1}^{d} (x_i - y_j)^2} \tag{2.2}$$

Both Manhattan distance and Euclidean distance are metrics or distance functions in $\mathbb{R}$.

If D is a set of discrete metric space, the discrete metric is defined by

$$\widehat{d_3}(x, y) = \begin{cases} 1 & if\ x \neq y \\ 0 & if\ x = y \end{cases} \tag{2.3}$$

2.2 Clustering algorithm

In this section, we describe a concept of clustering algorithm, k-means and k-medoids algorithms which are used to compare with the half-orbital extreme pole clustering algorithm.

Clustering algorithm is the procedure for organizing instances into clusters such that the instances in the same cluster are more similar to one another than the instances in other clusters. In other words, clustering algorithm tries to form instances into clusters with high intra-cluster similarity (low intra-cluster distances) and low inter-cluster similarity (high inter-cluster distances).

The words "high intra-cluster similarity" means that the instances in the same cluster have low intra-cluster distances. The intra-cluster distance of a cluster is measured in several ways, such as the summation of distances between every instance in the cluster and its center, the maximal distance between any pair of instances in the cluster, and the summation of variances in the cluster. While the intra-cluster distance is used to measure similarity in the cluster, inter-cluster distance is used to measure similarity between clusters. The inter-cluster distance can also be measured in various ways, such as the smallest distance between an instance in one cluster and an instance in the other cluster (single link), the largest distance between an instance in one cluster and an instance in the other cluster (complete link) and the average distance between an instance in one cluster and an instance in the other (average link).

The representative of each cluster formed by clustering algorithm is called the center. However, in different clustering algorithms, centers of clusters are computed in different methods. For example, k-means algorithm computes a center from mean value of all instances in a cluster and each center is called centroid. K-medoids algorithm selects one of instances as center and calls its center a medoid.

In this research, we used k-means and k-medoids algorithms to compare with our algorithm.

2.2.1 K-means algorithm

Although there are several k-means clustering algorithms [16], the concept of k-means algorithms is the same. All k-means algorithms aim to organize instances into k clusters, where k is the number of clusters that is initially given.

The definition of centroid is defined as follows.

Let $D = \{x_1, x_2, \dots , x_N\}$ be a set of the d-dimensional real-value vectors,

$x_i = (x_{i_1}, x_{i_2}, \dots , x_{i_d})$ be the i^{th} instance in D,

$C_1, C_2, \dots, C_k$ be the clusters in D where k is the number of clusters,

$c_t = (c_{t_1}, c_{t_2}, \dots, c_{t_d})$ be the centroid of the cluster C_t,

l_t be the number of instances in C_t for $t = 1,2, \dots, k$.

The components of centroid c_t are defined by

$$c_{t_j} = \frac{\sum_{i=1}^{l_t} x_{i_j}}{l_t} \tag{2.4}$$

for $j = 1,2, \dots, d$

This research used Lloyd's algorithm [9], which is a k-means clustering algorithm, as the comparative algorithm. Lloyd's algorithm is described as follows.

<u>Lloyd's algorithm</u>

```
INPUT: $k$ is the number of clusters,
       $D = \{x_1, x_2, \dots , x_N\}$ is a set of the $d$-dimensional real-value vectors,
       $x_i = (x_{i_1}, x_{i_2}, \dots , x_{i_d})$ be the $i^{\text{th}}$ instance in $D$
OUTPUT: A set of $k$ clusters
STEP 1 Choose randomly $k$ centroids $c_1, c_2, \dots, c_k$ from $D$ as centers of clusters
       $C_1, C_2, \dots, C_k$
STEP 2 For all instances $x_i$ for $i = 1,2, \dots, N$
       - Assign instance $x_i$ to a cluster $C_l$ where
         $\hat{d}(x_i, c_l) = \min_{j=1,2,\dots,k} \{\hat{d}(x_i, c_j)\}$.
STEP 3 Compute new centroids of clusters
STEP 4 Go back to STEP 2 until all instances do not change their groups.
```

<u>An Example of Lloyd's algorithm</u>

In this example, we organize the following instances into three clusters using Lloyd's algorithm.

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
Point	(2, 10)	(2, 5)	(8, 4)	(5, 8)	(7, 5)	(6, 4)	(1, 2)	(4, 9)

Initial centroids of clusters are given by $x_1 = (2,10)$, $x_4 = (5,8)$ and $x_7 = (1,2)$.

Start with calculating distances from centroids to each instance using Manhattan distance (2.1).

For instance x_1,

$$\hat{d}(x_1, x_1) = |x_{1_1} - x_{1_1}| + |x_{1_2} - x_{1_2}| = |2-2| + |10-10| = 0$$

$$\hat{d}(x_1, x_4) = |x_{1_1} - x_{4_1}| + |x_{1_2} - x_{4_2}| = |2-5| + |10-8| = 5$$

$$\hat{d}(x_1, x_7) = |x_{1_1} - x_{7_1}| + |x_{1_2} - x_{7_2}| = |2-1| + |10-2| = 9$$

The minimum distance from x_1 to each cluster is 0, so x_1 is assigned to the cluster that has x_1 as the centroid.

Iteration 1

	Point	(2, 10) Distance	(5, 8) Distance	(1, 2) Distance	Cluster
x_1	(2, 10)	0	5	9	1
x_2	(2, 5)	5	6	4	3
x_3	(8, 4)	12	7	9	2
x_4	(5, 8)	5	0	10	2
x_5	(7, 5)	10	5	9	2
x_6	(6, 4)	10	5	7	2
x_7	(1, 2)	9	10	0	3
x_8	(4, 9)	3	2	10	2

From above table,

Cluster 1: $\{x_1\}$, Cluster 2: $\{x_3, x_4, x_5, x_6, x_8\}$, and Cluster 3: $\{x_2, x_7\}$.

The new centroids of clusters are calculated.

For cluster 1, there is only one instance x_1, so the centroid remains the same.

For cluster 2, the centroid is $\left(\frac{(8+5+7+6+4)}{5}, \frac{(4+8+5+4+9)}{5}\right) = (6,\ 6)$.

For cluster 3, the centroid is $\left(\frac{2+1}{2}, \frac{5+2}{2}\right) = (1.5,\ 3.5)$.

Iteration 2

	Point	(2, 10) Distance	(6, 6) Distance	(1.5, 3.5) Distance	Cluster
x_1	(2, 10)	0	8	9	1
x_2	(2, 5)	5	5	2	3
x_3	(8, 4)	12	4	7	2
x_4	(5, 8)	5	3	8	2
x_5	(7, 5)	10	2	7	2
x_6	(6, 4)	10	4	5	2
x_7	(1, 2)	9	9	2	3
x_8	(4, 9)	3	5	7	1

From above table,

Cluster 1: $\{x_1, x_8\}$, Cluster 2: $\{x_3, x_4, x_5, x_6\}$, and Cluster 3: $\{x_2, x_7\}$.

Repeat algorithm until no changing.

The Lloyd's algorithm stops with

Cluster 1: $\{x_1, x_4, x_8\}$, Cluster 2: $\{x_3, x_5, x_6\}$, Cluster 3: $\{x_2, x_7\}$.

2.2.2 K-medoids algorithm

K-medoids algorithm is the clustering algorithm that also aims to partition instances into k clusters, where k is the number of clusters that is initially given, like k-means algorithm. The difference of these two algorithms is the center selecting scheme to form clusters. K-means algorithm uses mean value of instances in each cluster as a center. Therefore, the center of k-means algorithm is not need to be ones of instances. On the contrary, k-medoids algorithm uses one of the instances as a center called medoid. Thus, k-medoids algorithm has the condition to identify which instance should be the medoid in the next iteration.

The total cost value is the measurement for determining whether an instance should be a medoid in the next iteration. The computation of the total cost value is as follows.

Let $D = \{x_1, x_2, \dots , x_N\}$ be the set of the d-dimensional real-value vectors, $x_i = (x_{i_1}, x_{i_2}, \dots , x_{i_d})$ be the i^{th} instance in D,

$m_1, m_2, \dots, m_k$ be the medoids of clusters $C_1, C_2, \dots, C_k$.

1. Assume that clusters have been formed by assigning instances x_i to the clusters with medoid m_l if their distances are corresponding to the following equation:

$$\hat{d}(x_i, m_l) = \min_{\substack{j=1,2,\ldots,k \\ x_i \neq m_j \, \forall j}} \{\hat{d}(x_i, m_j)\}. \tag{2.5}$$

2. Compute the total cost by

$$total\ cost \ = \ \sum_{l=1}^{k} \sum_{x_i \in C_l} \hat{d}(x_i, m_l). \tag{2.6}$$

In order to select a new medoid, the current medoid is tentatively replaced by instances that are not currently medoids. The total cost value for each replacement is then computed. The instance that gives the minimum total cost value will actually replace the current medoid in next iteration.

This research study used Partition Around Medoids algorithm (PAM) [10], which is one of k-medoid clustering algorithms, as a comparative algorithm. PAM algorithm is described as follows.

<u>Partition Around Medoids (PAM) algorithm</u>

```
INPUT: $k$ is the number of clusters,
       $D = \{x_1, x_2, \ldots , x_N\}$ is a set of the $d$-dimensional real-value vectors,
       $x_i = (x_{i_1}, x_{i_2}, \ldots , x_{i_d})$ is the $i^{\text{th}}$ instance in $D$
OUTPUT: A set of $k$ clusters
STEP 1 Choose randomly $k$ medoids $m_1, m_2, \ldots, m_k$ from $D$ as centers of clusters
        $C_1, C_2, \ldots, C_k$
STEP 2 For all instances $x_i$ for $i = 1,2, \ldots, N$
        - Assign instance $x_i$ to a cluster $m_l$ according to equation (2.5)
STEP 3 Calculate the current total cost value
STEP 4 For all medoids $m_l$ for $l = 1,2, \ldots, k$
        - For all instances $x_i$ for $i = 1,2, \ldots, N$ which $x_i$ is not medoids
            - Swap $m_l$ and $x_i$ and compute the total cost
STEP 5 Replace $m_l$ with $x_i$ which gives lowest total cost and that total cost value is
        lower than current total cost value.
STEP 6 Go back to STEP 2 until there is no change in the medoids.
```

<u>An Example of PAM algorithm</u>

In this example, we organize the following instances into 2 clusters using PAM algorithm.

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
Point	(2, 6)	(3, 4)	(3, 8)	(4, 7)	(6, 2)	(6, 4)	(7, 3)	(7, 4)	(8, 5)	(7, 6)

STEP 1 Choose randomly two medoids $m_1 = x_2 = (3, 4)$ and $m_2 = x_8 = (7, 4)$ as centers of clusters $C_1$and C_2

STEP 2 For all instances x_i for $i = 1,2, \dots ,10$

- Assign instance x_i to a cluster m_l according to equation (2.5)

	Point	(3, 4) Distance	(7, 4) Distance	Cluster
x_1	(2, 6)	**3**	7	1
x_3	(3, 8)	**4**	8	1
x_4	(4, 7)	**4**	6	1
x_5	(6, 2)	5	**3**	2
x_6	(6, 4)	3	**1**	2
x_7	(7, 3)	5	**1**	2
x_9	(8, 5)	6	**2**	2
x_{10}	(7, 6)	6	**2**	2

Since $x_1, x_3$and x_4 are closer to medoid x_2, they are formed into one cluster. The remaining instances, x_5, x_6, x_7, x_9 and x_{10}, are closer to medoid x_8, thus, they are formed into the other cluster.

Cluster 1: $\{x_1, x_2, x_3, x_4\}$ and Cluster 2: $\{x_5, x_6, x_7, x_8, x_9 , x_{10}\}$.

STEP 3 Calculate the current total cost value

So total cost $= (3 + 4 + 4) + (3 + 1 + 1 + 2 + 2) = 11 + 9 = 20$.

STEP 4 For all medoids m_l for $l = 1,2, \dots , k$

- For all instances x_i for $i = 1,2, \dots , N$ which x_i is not medoids
    - Swap m_l and x_i and compute the total cost

For $m_1 = x_2$, the total cost value is calculate for each replacement of m_1 by $x_1, x_3, x_4, x_5, x_6, x_7, x_9$ and x_{10}.

For $m_2 = x_8$, the total cost value is calculate for each replacement of m_2 by $x_1, x_3, x_4, x_5, x_6, x_7, x_9$ and x_{10}.

For this example, we replace medoids $m_2 = x_8 = (7, 4)$ with $x_7 = (7, 3)$ while $m_1 = x_2$ stays the same. Then, we compute the total cost

	Point	(3, 4) Distance	(7, 3) Distance	Cluster
x_1	(2, 6)	**3**	8	1
x_3	(3, 8)	**4**	9	1
x_4	(4, 7)	**4**	7	1
x_5	(6, 2)	5	**2**	2
x_6	(6, 4)	3	**2**	2
x_8	(7, 4)	4	**1**	2
x_9	(8, 5)	6	**3**	2
x_{10}	(7, 6)	6	**3**	2

Thus, total cost of x_7 which replaces m_2

$$= (3 + 4 + 4) + (2 + 2 + 1 + 3 + 3) = 11 + 11 = 22.$$

After all total cost values are computed, if there is no the total cost value that is lower than the current total cost, the algorithm will stop. In other case, if there are the total cost values that are lower than the current total cost, the pair of instance and medoid that gives the minimum total cost value will be selected. The selected instance is set as the current medoid in next iteration.

2.3 Performance measure

There are several performance measures to evaluate clustering algorithms. Total variance is one of the measures which are used in comparison with k-means and k-medoids algorithms. The detail of total variance is explained as follows.

2.3.1 Total Variances

Total variance is used as a measure to compare performance between two clustering algorithms. The total variance is calculated from the summation of variance of each cluster. Because a variance value from a cluster shows how far the set of instances in the cluster disperse from its centroid, the total variance can imply dispersion and aggregation of all instances in the dataset.

Let $D = \{x_1, x_2, \ldots, x_N\}$ be a set of the d-dimensional real-value vectors,

$x_i = (x_{i_1}, x_{i_2}, \ldots, x_{i_d})$ be the i^{th} instance in D,

$C_1, C_2, \ldots, C_k$ be the clusters in D where k is the number of cluster,

$c_t = (c_{t_1}, c_{t_2}, \ldots, c_{t_d})$ be the centroid of the cluster C_t; c_{t_i} is computed from equation (2.4)

l_t be the number of instances in C_t for $t = 1,2, \ldots, k$.

The variance of cluster C_t is then defined by

$$var\,(C_t) = \frac{\sum_{i=1}^{l_t} \sum_{j=1}^{d} \left(x_{i_j} - c_{t_j}\right)^2}{l_t - 1}. \tag{2.7}$$

Hence, the total variance of k clusters is defined by

$$\text{Total variance} = \sum_{t=1}^{k} var(C_t) \tag{2.8}$$

<u>The example of variance calculation</u>

This example demonstrates how to compute the variance of a cluster.

Let x_1, x_2, and x_3 be the instances in a cluster where $x_1 = (1,1)$, $x_2 = (1,2)$, $x_3 = (3,3)$.

The centroid of a cluster is $C = (c_1, c_2)$ where

$$c_1 = \frac{\sum_{i=1}^{3} x_{i_1}}{3} = \frac{1+1+3}{3} = \frac{5}{3},$$

$$c_2 = \frac{\sum_{i=1}^{3} x_{i_2}}{3} = \frac{1+2+3}{3} = \frac{6}{3} = 2.$$

Thus, the variance is computed by the equation (2.7).

$$var\,(C_t) = \frac{\sum_{i=1}^{l} \sum_{j=1}^{d} \left(x_{ij} - c_{tj}\right)^2}{l_t - 1}$$

$$= \frac{\sum_{i=1}^{3} \left(x_{i1} - \frac{5}{3}\right)^2}{3 - 1} + \frac{\sum_{i=1}^{3} (x_{i2} - 2)^2}{3 - 1}$$

$$= \frac{\left(\left(1 - \frac{5}{3}\right)^2 + \left(1 - \frac{5}{3}\right)^2 + \left(3 - \frac{5}{3}\right)^2\right)}{2} + \frac{((1-2)^2 + (2-2)^2 + (3-2)^2)}{2}$$

$$= \frac{\left(\frac{4}{9}+\frac{4}{9}+\frac{16}{9}\right)}{2} + \frac{(1+0+1)}{2}$$

$$= \frac{7}{3}$$

2.4 Pre-processing

First, all instances in every dataset must be normalized. This step is very important because the values of instances in different scales will be transformed into the same scale. Specifically, we apply the normalization technique to scale all numeric values into the range $[0, 1]$ as seen in the following formula:

$$z_{i_j} = \frac{x_{i_j} - \min_k\{x_{i_k}\}}{\max_k\{x_{i_k}\} - \min_k\{x_{i_k}\}} \tag{2.9}$$

where z_{i_j} is the j^{th} normalized component of the i^{th} instance.

2.5 Literature review

In this part, we explain the literature that induces the idea of the farthest pair, which is the main concept of this work. We begin with describing the multi-attributed frame [14] and the multi-attributed lens [15] which both used the idea of farthest pair. Although both papers are classification-related, the idea of farthest pair is also applicable to clustering.

2.5.1 Multi-attributed frame

The idea of the multi-attributed frame is proposed in [14]. This paper suggests the new approach to decision tree, which is one of the algorithms in classification. This paper uses an idea of the farthest pair, which is a pair of two instances that have the maximum distance, to limit the considered region.

The first step is finding the farthest pair which is called extreme poles. After the farthest pair is obtained, the vector core is created from this pair. Consequently, there are two planes that are perpendicular to the vector core at the borders and the region of instances is partitioned into three sub regions: right region, middle region, and left region. (See Figure 1)

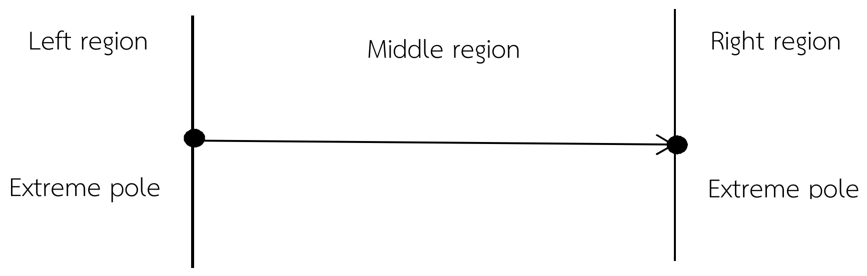


Figure 1 Three regions are partitioned into a frame by a vector core.

Since the vector core is generated from the two extreme poles that have the largest distance, this guarantees that all instances lie in the middle region. After that, all instances are projected onto this core. Hence, several attributes are reduced to single attribute, and then the splitting point is found so that all instances in the middle region will be divided into specified class and unspecified class. The algorithm is conducted recursively with the unspecified class until the stopping criteria are met.

The concept of the farthest pair can limit the considered region by choosing the same type of target class of the farthest pair. For example, suppose that target classes of all instances are positive and negative. If both poles are the positive, there are no positive instances that lie in right and left regions: all instances in right and left regions are negative instances. On the other hand, if both poles are negative, there is also no negative instance that lies in right and left regions. Moreover, if the target classes of two extreme poles are different, the target class in right and left regions can be still guaranteed. By the properties of the farthest pair, the target class of instances in right region is not the same as the target class of right pole. Similarly, the target class of instances in left region is not the same as the target class of left pole. In other words, we can always guarantee the target class of all instances in right and left regions. So there is only the middle region left to be considered.

2.5.2 Multi-attributed lens

The idea of the farthest pair that is initially proposed in the multi-attributed frame decision tree is also applied in [15] which also focused on the decision tree. This paper improved the concept of considered regions from three regions to two regions.

Although the multi-attributed frame lens and multi-attributed lens are both decision tree algorithms using multi-attributed concept, there are some different points between these two papers. Even though both algorithms start by finding the farthest pair and generating the vector core, the multi-attributed lens requires that the target classes of extreme poles must be the same type. Furthermore, the considered regions are changed from the frame to the lens as explained in the following detail. After generating the vector core, the multi-attributed lens creates two balls by setting the extreme poles as the centers of each ball and the length of vector core as their radiuses. (See Figure 2)

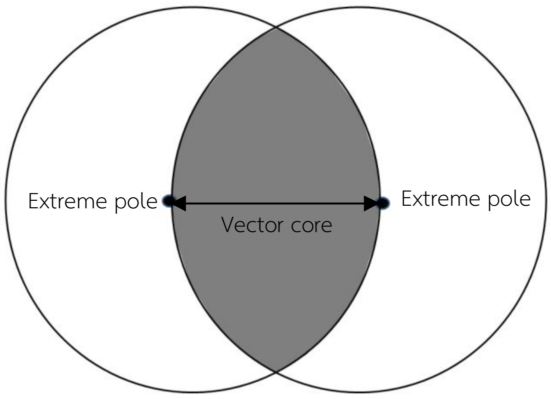


Figure 2 All instances having the same target class with the extreme poles lie in the lens.

Since the vector core, whose length is the maximum distance of two instances, is set as the radius of the two balls, all instances with the same target class as the extreme poles do not lie away from the extreme poles further than the length of the vector core. In other words, all instances with the same target class as the extreme poles lie in the intersection of two balls, which is called lens. The shaded region in Figure 2 indicates an example of a lens. Therefore, the regions are divided as the region inside the lens and the one outside the lens. Moreover, all instances lying outside the lens are immediately identified with their target class.

After identifying the target class of all instances outside lens, instances inside the lens are projected onto the vector core, and the splitting point can be found so that all instances inside the lens should be partitioned into specified instances and unspecified instances. The algorithm keeps finding the farthest pair of the unspecified instances and repeats the process until the stopping criteria are met.

2.6 Background concept

Although both ideas from the papers, which are multi-attributed frame and multi-attributed lens, are used in classification, the concept of the farthest pair is also applicable to clustering algorithm as well. In this section, we state how the farthest pair is applied in clustering and describe the definitions of dispersion which includes Sturges' formula used in our algorithm.

2.6.1 The farthest pair in clustering

The concept of farthest pair is applied in our clustering algorithm due to the following reasons. Since the farthest pair is a pair of instances that have the maximum distance, the clustering of entire instances can be fallen in two possible cases. The first case is that all can be in the same group if the dispersion of instances along the vector core is homogeneous. The other case is that if the dispersion along the vector core is nonhomogeneous, they should be in different groups. In case of nonhomogeneous, the farthest pair is the first two considered instances lied in different groups. Moreover, they are considered as the main points on partitioning the rest of instances.

2.6.2 Dispersion

While we discuss about applying the farthest pair concept in our clustering algorithm in the previous section, we have mentioned the dispersion along the vector core used for specifying the cases of instances whether they are homogeneous or nonhomogeneous. In this section, we formally define the words *dispersion*, *homogeneous* and *nonhomogeneous*.

Since the objective of clustering algorithm is to partition instances into groups, the dispersion will be used in our clustering algorithm as the criterion to decide whether all instances should be partitioned. This research study uses histogram as the tool for inspecting the dispersion of instances by the following method. First, the farthest pair is found and set as the extreme poles. Then, the vector core is constructed from these poles. After that, the histogram is created along the vector core. The pattern of a chain of bins collecting from the frequency of instances in this histogram will be called the *dispersion* along the vector core. The dispersion along the vector core indicates whether a group of instances is *homogeneous* or *nonhomogeneous*. A *homogeneous* group means all instances should belong in the same group, therefore the partitioning does not occur in this group. The examples of dispersion in this case are normal curve, positively skewed curve or negatively skewed curve as shown in Figure 3.

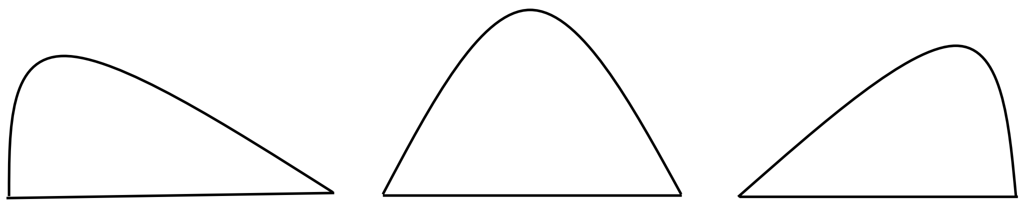


Figure 3 Positively skewed curve, normal curve and negatively skewed curve respectively

On the other hand, a *nonhomogeneous* group suggests that the instances should be partitioned into the different groups. Hence, the ball with an appropriate radius of which center is one of extreme poles is built. The instances are partitioned into two groups which are groups of instances inside the ball and outside the ball. Figure 4 shows the examples of dispersion in this case.

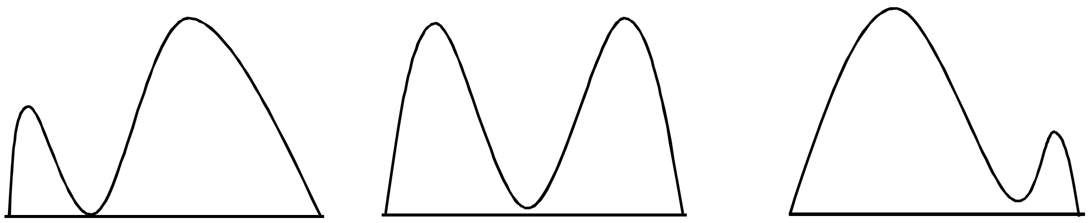


Figure 4 The examples of dispersion in nonhomogeneous

2.6.2.1 Sturge's formula

In order to form a histogram, we construct a set of non-overlapping intervals called bins and count the number of instances in each bin. To compare the frequency of bins, the bins should have the same width. In our study, the number of bins is determined by Sturges' formula, which was first proposed by Herbert A. Sturges [17].

$$n = \lceil \log_2 |S| + 1 \rceil \tag{2.7}$$

where n is the number of bins and S is the set of unspecified instances.

Sturges's formula is widely recommended in many introductory statistics textbooks and is often used in statistical packages in several programs as a default method. Scott [18] interpreted Sturges' formula by using the concept of binomial coefficient in 1992 as follows. The first step is constructing the histogram with n bins with the equal width. In the second step, we assume the frequency of the k^{th} bin as a binomial coefficient $\binom{n-1}{k}$ for $k = 0, 1, 2, \dots, n-1$. As k increases, the histogram is tended to be the shape of normal distribution. Furthermore, the total number of instances is

$$|S| = \sum_{k=0}^{n-1} \binom{n-1}{k} = (1+1)^{n-1} = 2^{n-1} \tag{2.8}$$

by the binomial expansion. We find that equation (2.7) and (2.8) are corresponding.

CHAPTER III
HALF-ORBITAL EXTREME POLE CLUSTERING ALGORITHM

This chapter proposes the half-orbital extreme pole clustering algorithm (HOEP algorithm). In order to understand the HOEP algorithm, first, the notations are introduced. After that, the HOEP algorithm will be proposed and described. Moreover, the pseudo code and flowchart of algorithm are provided later in this chapter.

3.1 The notations for Half-Orbital Extreme Pole algorithm

Let

- N be the number of all instances;
- $x_i = \left(x_{i_1}, x_{i_2}, \dots, x_{i_d}\right)$ be the i^{th} instance, which is a d-dimensional vector of real numbers for all $i = 1, 2, \dots, N$;
- $D = \{x_1, x_2, \dots, x_N\}$ be the set of all instances;
- S be the set of the considered instances;
- $\bar{A} = \left[a_{ij}\right]$ be the distance matrix where a_{ij} is the Euclidean distance from the i^{th} instance to the j^{th} instance;
- p_1 and p_2 be the farthest pair of all instances called extreme poles;
- $\vec{v}$ be a vector core generated by the extreme poles;
- α be the width of intervals used for constructing the histogram bins;
- n be the number of intervals with the length α used for constructing the histogram bins;
- $B(c; r)$ be the ball (orbit) with a center c and a radius r;
- f_i be the number of instances that lie between $B(c; (i-1)\alpha)$ and $B(c; i\alpha)$;
- f_i^* be the transformed value of f_i into the range $[0, 1]$;
- γ be the parameter for determining the cluster splitting point.

3.2 Half-Orbital Extreme Pole algorithm

INPUT: $D = \{x_1, x_2, \dots, x_N\}$ is the set of d-dimension real vector, where

$x_i = \left(x_{i_1}, x_{i_2}, \dots, x_{i_d}\right)$ is the i^{th} instance,

parameter γ.

OUTPUT: Clusters of instances

First, HOEP algorithm sets $S = D$. The HOEP algorithm runs on S instead of D so that the algorithm should stop when the number of considered instances is too small.

Second, the distance matrix $\bar{A} = [a_{ij}]$ is created where a_{ij} is the Euclidean distance from x_i and x_j. The Euclidian distance is calculated as indicated in the formula (3.1):

$$a_{ij} = \sqrt{\sum_{t=1}^{d}\left(x_{i_t} - x_{j_t}\right)^2} \tag{3.1}$$

where $x_i = \left(x_{i_1}, x_{i_2}, \dots, x_{i_d}\right)$ and $x_j = \left(x_{j_1}, x_{j_2}, \dots, x_{j_d}\right)$.

Third, the extreme poles p_1 and p_2 are identified by the index of the maximum value of elements in the distances matrix $\bar{A}$. In other word, if $a_{i'j'}$, which is an element in the distance matrix $\bar{A}$, has the largest value among of all elements in $\bar{A}$, then $x_{i'}$ and $x_{j'}$ are set to be the extreme poles p_1 and p_2 respectively.

Fourth, the vector core $\vec{v}$ is generated from the extreme poles, p_1 and p_2, with magnitude $\|\vec{v}\|$, where

$$\|\vec{v}\| = \sqrt{\sum_{i=1}^{d}(p_{1i} - p_{2i})^2} \tag{3.2}$$

Next, the vector core $\vec{v}$ is divided into n equal width intervals. So the length of intervals α is

$$\alpha = \frac{\|\vec{v}\|}{n} \tag{3.3}$$

where the number of intervals n is calculated from Sturges' formula (Sturges 1926)

$$n = \lceil \log_2 |S| + 1 \rceil. \tag{3.4}$$

This formula has already been explained in previous chapter.

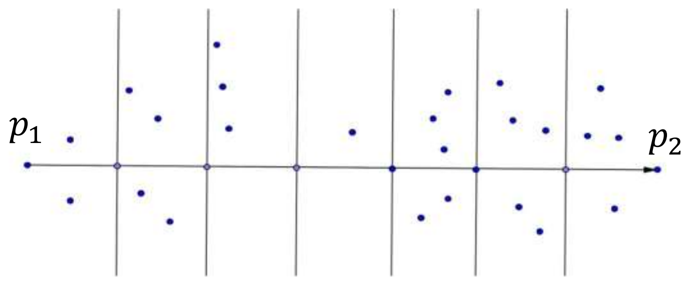


Figure 5 The vector core is divided into $\boldsymbol{n}$ intervals of equal width $\boldsymbol{\alpha}$.

Fifth, $\mathbf{p_1}$ is chosen as the center of the balls (orbits) which are constructed with different radiuses, each of which is a multiple of α (See Figure 6). Then, instances in each layer of a ball are counted to creat87e a histogram. f_i^* is the transformed frequency for the i^{th} layer. If f_i is the number of instances that lie between $B(c;(i-1)\alpha)$ and $B(c;i\alpha)$, f_i^* is calculated from one of two transformed functions:

the first transformed function

$$f_i^* = g_1(f_i) = \frac{f_i}{|S|} \tag{3.5}$$

the second transformed function

$$f_i^* = g_2(f_i) = \frac{f_i - \min_j f_j}{\max_j f_j - \min_j f_j}. \tag{3.6}$$

Thus, f_i^* is the normalized value of f_i into the range $[0,1]$.

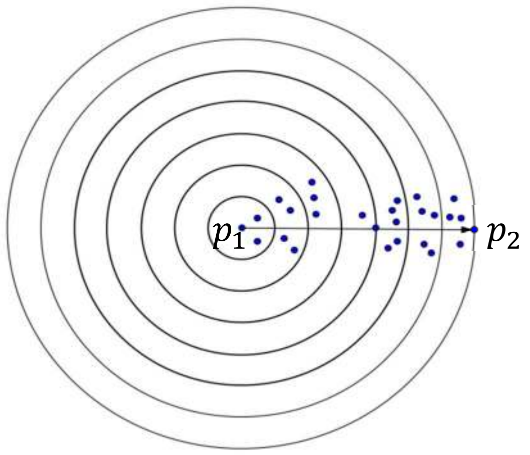


Figure 6 $\boldsymbol{p_1}$ is chosen as the center of balls with several determined radiuses.

In order to partition instances into two groups, the splitting point is determined by the following method. In the beginning, the parameter γ is set as the indicator for determining a layer with a splitting point. After that, the j^{th} layer contains a splitting point if $f_j^* < \gamma$, $f_{j+1}^* > \gamma$ and $f_r^* > \gamma$ for all $r < j$. Figure 7 illustrates an example of such layer containing a splitting point.

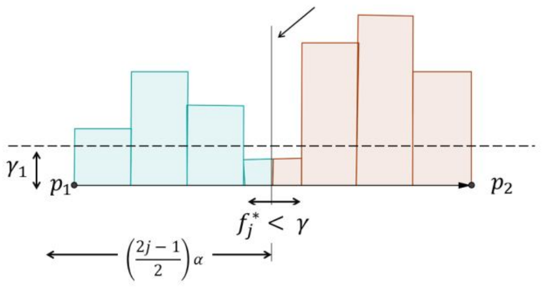


Figure 7 The desirable layer containing a splitting point is specified by parameter gamma.

Consequently, the midpoint of the j^{th} layer is set as the splitting point and the distance from selected pole p_1 to the splitting point is $\left(\frac{2j-1}{2}\right)\alpha$. This value is set to be the radius of a ball with p_1 as its center. In other words, the ball $B\left(p_1; \frac{2j-1}{2}\alpha\right)$ is created. Therefore, the instances are partitioned into two groups which are instances inside the ball and outside the ball. (See Figure 8) The instances inside the ball form a cluster.

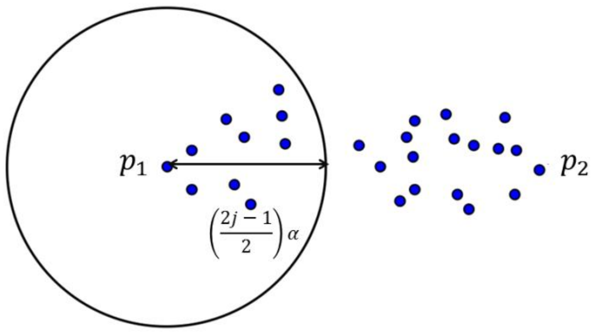


Figure 8 The ball $\boldsymbol{B\left(p_1; \frac{2j-1}{2}\alpha\right)}$ is created and all instances are divided into two groups which are instances inside and outside the ball.

Then, the algorithm repeats on instances outside the ball if the number of instances that are outside the ball is not larger than ninety percent of the size of input dataset. Otherwise, the algorithm terminates.

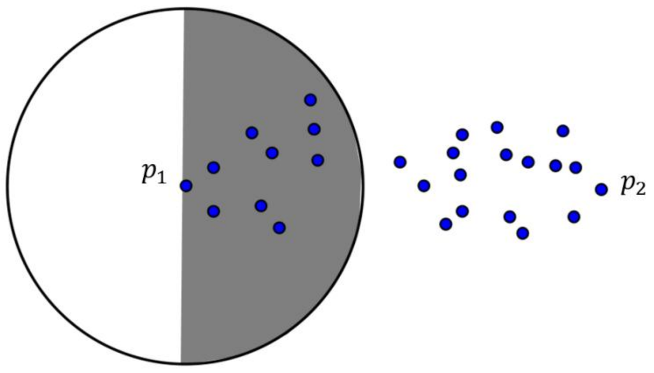


Figure 9 All instances in the ball lie on a half of ball with side along vector core.

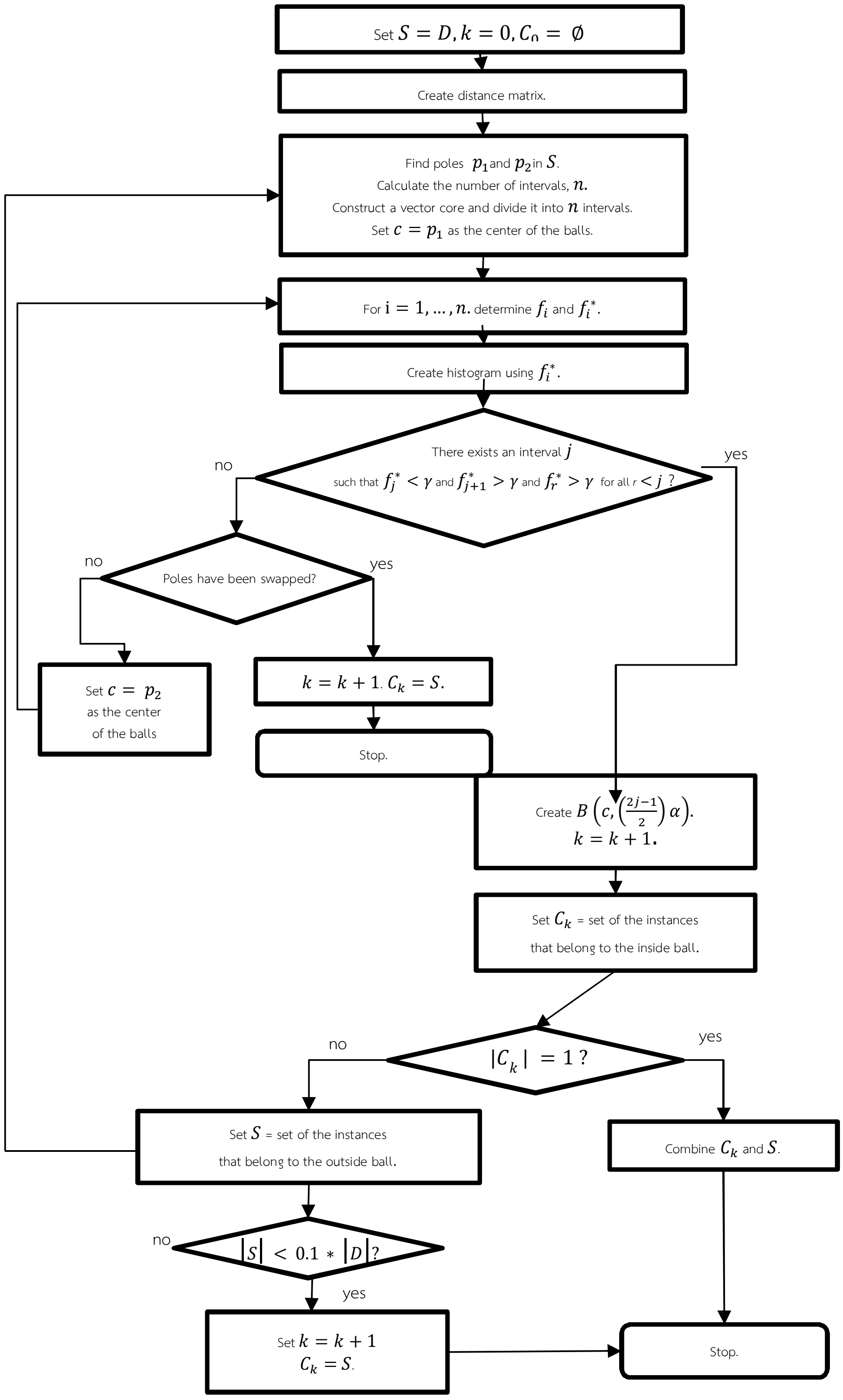
In case such a splitting point does not exist, the center of the balls is switched to the other extreme pole and new layers are created. A histogram is reconstructed and a splitting point is identified in a similar manner. If the algorithm still cannot partition the instances, then all instances are assigned to one group and the algorithm is terminated.

According to HOEP algorithm, when we run our algorithms several times with the same parameter γ, the final clusters do not change. In the beginning of the algorithm with the same dataset, it is obvious that the first farthest pairs that are found in different runs are the same pair. Therefore, the bins that correspond with splitting criterion in different runs are the same bin. Thus, the balls that are created to partition instances in the first time are also the same center and the same radiuses. Hence, the instances that are partitioned by these balls are the same. Thus, in the next iteration, the results from different runs do not change. Therefore, HOEP gives the same final clusters in different runs. Moreover, in the beginning of our algorithm, the number of clusters is not specified.

The pseudo code and flowchart of HOEP algorithm are provided here.

3.3 The pseudo code of HOEP algorithm

```
INPUT: $D = \{x_1, x_2, \dots, x_N\}$ is the set of $d$-dimension real vector, where
    $x_i = \left(x_{i_1}, x_{i_2}, \dots, x_{i_d}\right)$ is the $i^{\text{th}}$ instance,
    parameter $\gamma$.
OUTPUT: Clusters of instances: $C_1, C_2, \dots, C_k$
STEP 1: $S = D, k = 0$ and $C_0 = \emptyset$
STEP 2: Create distance matrix
STEP 3: Find extreme poles $p_1$ and $p_2$ in $S$
STEP 4: Construct a vector core $\vec{v}$, calculate the number of intervals $n$, and divide it into $n$ intervals
STEP 5: Set $c = p_1$ as the center of the balls
STEP 6: For $i = 1, \dots, n.$ determine $f_i$ and $f_i^*$
STEP 7: If there exists an interval $j$ such that $f_j^* < \gamma$ and $f_{j+1}^* > \gamma$ and $f_r^* > \gamma$ for all $r < j$ {
            - Create ball $B\left(c, \left(\frac{2j-1}{2}\right)\alpha\right)$
            - $k = k + 1$
            - Set $C_k$ = set of the instances that belong to the inside ball
            - If $|C_k| = 1$ {
                  - Combine $C_k$ and $S$
                  - Algorithm stops
            }else{
                  - Set $S$ = set of the instances that belong to the outside ball
                  - If $|S| < 0.1 \cdot |D|$ {
                        - $k = k + 1$
                        - $C_k = S$
                        - Algorithm stops
                  }else{
                        - Go back to Step 3
                  }
            }
      }else{
            - If poles have been swapped {
                  - $k = k + 1$
                  - Set $C_k = S$
                  - Algorithm stops
            }else{
                  - Set $c = p_2$ as the center of the balls
                  - Go back to Step 6
            }
      }
```



Set $S = D, k = 0, C_0 = \emptyset$
Create distance matrix.
Find poles p_1 and p_2 in S.
Calculate the number of intervals, n.
Construct a vector core and divide it into n intervals.
Set $c = p_1$ as the center of the balls.
For $i = 1, \dots, n$. determine f_i and f_i^*.
Create histogram using f_i^*.
There exists an interval j such that $f_j^* < \gamma$ and $f_{j+1}^* > \gamma$ and $f_r^* > \gamma$ for all $r < j$?
no
yes
Poles have been swapped?
no
yes
Set $c = p_2$ as the center of the balls
$k = k + 1.\ C_k = S.$
Stop.
Create $B\left(c, \left(\frac{2j-1}{2}\right)\alpha\right)$. $k = k + 1$.
Set C_k = set of the instances that belong to the inside ball.
$|C_k| = 1$?
no
yes
Set S = set of the instances that belong to the outside ball.
Combine C_k and S.
$|S| < 0.1 * |D|$?
no
yes
Set $k = k + 1$ $C_k = S$.
Stop.

Figure 10 The flowchart of HOEP algorithm

CHAPTER IV
EXPERIMENTS AND RESULTS

In this chapter, we describe our experiments and compare our results with k-means and k-medoids algorithms using the average value of total variance. In the experiment, Half-Orbital Extreme Pole is written in R and performed on a PC with Intel(R) Core(TM) i5-2430M 2.40 GHz CPU and 6GB of RAM on Microsoft Windows 7 Operating System.

4.1 Dataset description

In this experiment, we use 4 datasets which are iris, seed, E.coli and wine datasets from UCI repository.

4.1.1 Iris dataset

The iris dataset consists of three species of iris plants which are Setosa, Virginica, and Versicolor. The iris dataset consists of 150 instances (50 instances for each) and 4 attributes, which are sepal length, sepal width, petal length, and petal width in centimeters, with no missing value. The target class of iris dataset is excluded since it is irrelevant to the clustering task. The description of the iris dataset is shown in Table 1 and Figure 11 shows a scatter plot of the iris dataset which is plotted between two principal components axes.

Table 1 The statistical information of iris dataset

	Min	Max	Mean	SD
sepal length	4.3	7.9	5.84	0.83
sepal width	2.0	4.4	3.05	0.43
petal length	1.0	6.9	3.76	1.76
petal width	0.1	2.5	1.20	0.76

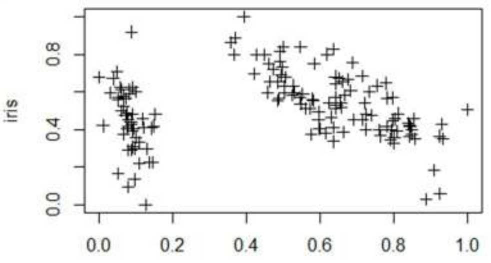


Figure 11 Iris dataset

4.1.2 Seed dataset

The seed dataset is the description of three types of wheat kernels which are Kama, Rosa and Canadian. The seed dataset consists of 210 instances (70 instances for each type) and 7 attributes, which are area, perimeter, compactness, length of kernel, width of kernel, asymmetry coefficient and length of kernel groove, with no missing value. The target class of seed dataset is also ignored. The information of the seed dataset is shown in Table 2 and Figure 12 shows a scatter plot of the seed dataset which is plotted between two principal components axes.

Table 2 The statistical information of seed dataset

	Min	Max	Mean	SD
area	10.59	21.18	14.85	2.909699
perimeter	12.41	17.25	14.56	1.305959
compactness	0.8081	0.9183	0.871	0.023629
length of kernel	4.899	6.675	5.629	0.443063
width of kernel	2.63	4.033	3.259	0.377714
asymmetry coefficient	0.7651	8.456	3.7002	1.503557
length of kernel groove	4.519	6.55	5.408	0.491481

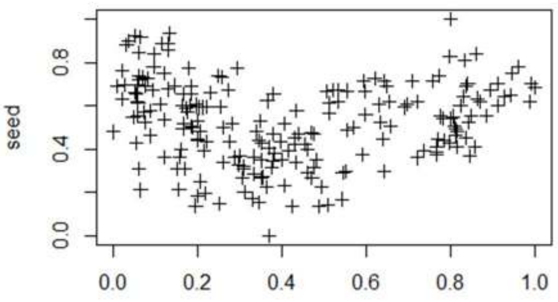


Figure 12 Seed dataset

4.1.3 E.coli dataset

The E.coli dataset consists of 336 instances, and 8 attributes (7 predictive, 1 name) with no missing value. The name attribute and the target class are ignored. Thus, there are 7 attributes left. The information of this dataset is shown in Table 3 and Figure 13 shows a scatter plot of the E.coli dataset which is plotted between two principal components axes.

Table 3 The statistical information of E.col dataset

	Min	Max	Mean	SD
mcg	0	0.89	0.5001	0.194634
gvh	0.16	1	0.5	0.148157
lip	0.48	1	0.4955	0.088495
chg	0.5	1	0.5015	0.027277
aac	0	0.88	0.5	0.122376
alm1	0.03	1	0.5002	0.215751
alm2	0	0.99	0.4997	0.209411

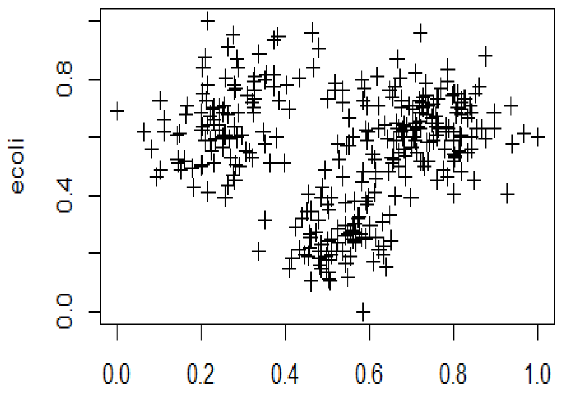


Figure 13 E.coli dataset

4.1.4 Wine dataset

The wine dataset is the description of three different types of wines. The chemical analysis gives the quantities of thirteen compositions which are found in each of three different cultivars of wines. The wine dataset consists of 178 instances, and 13 attributes with no missing value. The statistical information of wine dataset is shown in the following table and Figure 14 shows a scatter plot of the wine dataset which is plotted between two principal components axes.

Table 4 The statistical information of wine dataset

	Min	Max	Mean	SD
Alcohol	11.03	14.83	13	0.811827
Malic acid	0.74	5.8	2.336	1.117146
Ash	1.36	3.23	2.367	0.274344
Alcalinity of ash	10.6	30	19.49	3.339564
Magnesium	70	162	99.74	14.28248
Total phenols	0.98	3.88	2.295	0.625851
Flavanoids	0.34	5.08	2.029	0.998859
Nonflavanoid phenols	0.13	0.66	0.3619	0.124453
Proanthocyanins	0.41	3.58	1.591	0.572359
Color intensity	1.28	13	5.058	2.318286
Hue	0.48	1.71	0.9574	0.228572
OD280/OD315 of diluted wines	1.27	4	2.612	0.7099904
Proline	278	1680	746.9	314.9075

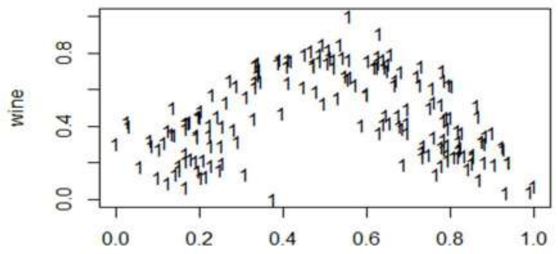


Figure 14 Wine dataset

The following table summarizes information of all datasets.

Table 5 Dataset Description

Dataset	Attributes	Instances
Iris	4	150
Seed	7	210
E.coli	7	336
Wine	13	178

4.2 Results of the experiments

In this section, we compare the total variances of our clustering algorithm with k-means and k-medoids algorithm. Since our algorithm gives the same final clusters in different runs, the total variances in different runs are the same value. We run 100 rounds of each compared algorithms and use the average values of total variances to compare with our results.

At the beginning, we set different values of gamma by varying from 0.05 to 1 with 0.05 increments. Then, we run HOEP algorithm only once for each gamma value. We compare our algorithm with k-means and k-medoids algorithms by setting the number of clusters of compared algorithms equal to the number of clusters from our algorithm. We run 100 rounds of each compared algorithms to compare with our result in each run of gamma value. Finally, we choose the gamma value that gives the minimum total variance to be the appropriate gamma value of each dataset. The results of experiments are shown as follows.

Iris dataset

The following table shows the number of clusters and the total variance from HOEP algorithm. The values of gamma are varied from 0.05 to 1 with 0.05 increments. The result shows that the gamma values that give the minimum total variance in HOEP algorithm for $g_1(f_i)$ is 0.25 and for $g_2(f_i)$ are 0.35 and 0.40.

Table 6 Results of iris dataset from HOEP algorithm

	$f_i^* = g_1(f_i) = \frac{f_i}{\lvert S \rvert}$		$f_i^* = g_2(f_i) = \frac{f_i - \min_j f_j}{\max_j f_j - \min_j f_j}$	
γ	The number of clusters	Total variance HOEP	The number of clusters	Total variance HOEP
0.05	1	0.274255	1	0.2742545
0.10	3	0.139462	1	0.2742545
0.15	2	0.270116	1	0.2742545
0.20	5	0.257964	5	0.1485446
0.25	3	**<u>0.137439</u>**	5	0.1485446
0.30	2	0.139887	6	0.2736472
0.35	1	0.274255	3	**<u>0.1394619</u>**
0.40	1	0.274255	3	**<u>0.1394619</u>**
0.45	1	0.274255	3	0.1433193
0.50	1	0.274255	4	0.1440115
0.55	1	0.274255	4	0.1440115
0.60	1	0.274255	4	0.1440115
0.65	1	0.274255	10	0.1926997
0.70	1	0.274255	2	0.2701162
0.75	1	0.274255	6	0.248374
0.80	1	0.274255	6	0.248374
0.85	1	0.274255	6	0.248374
0.90	1	0.274255	4	0.2418484
0.95	1	0.274255	5	0.2450899
1.00	1	0.274255	1	0.2742545

Table 7 shows the results from HOEP comparing with ones from k-means and k-medoids algorithms. The number of clusters from the result of HOEP algorithm is set as the predetermined value in the initial step of k-means and k-medoids algorithms. Therefore, we compare the total variance of the same number of clusters in each algorithm. The results from iris dataset show that the total variances from

HOEP are lower than total variances from k-means and k-medoids algorithms in both of normalized functions.

Table 7 Comparison of the total variance of iris dataset

f_i^*	γ	The number of clusters	Total variance		
			HOEP	k-means	k-medoids
$g_1(f_i)$	0.25	3	0.137439	0.1483595	0.1398882
$g_2(f_i)$	0.35	3	0.1394619	0.1496138	0.1398587
	0.40	3	0.1394619	0.1466788	0.1399226

Although the HOEP algorithm can definitely partition instances in class Setosa after combining two final clusters, the HOEP cannot separate instances in class Virginica from Versicolor. Because, in the third iteration, there are only one instance which is one of extreme poles is partitioned from histogram, this means that this pole is not close to some of the remaining instances enough to form another cluster. Therefore, the algorithm terminates.

Seed dataset

Table 8 shows the number of clusters and the total variance of the seed dataset from HOEP algorithm. The values of gamma are varied from 0.05 to 1 with 0.05 increments. The result shows that the gamma values that give the minimum total variance in HOEP algorithm for $g_1(f_i)$ is 0.10 and for $g_2(f_i)$ are 0.10, 0.15, 0.20, 0.25, and 0.30.

Table 8 Results of seed dataset from HOEP algorithm

	$f_i^* = g_1(f_i) = \frac{f_i}{\lvert S \rvert}$		$f_i^* = g_2(f_i) = \frac{f_i - \min_j f_j}{\max_j f_j - \min_j f_j}$	
γ	The number of clusters	Total variance HOEP	The number of clusters	Total variance HOEP
0.05	2	0.418124	1	0.4237425
0.10	4	**<u>0.342573</u>**	2	**<u>0.4181241</u>**
0.15	3	0.386164	2	**<u>0.4181241</u>**
0.20	1	0.423743	2	**<u>0.4181241</u>**
0.25	1	0.423743	2	**<u>0.4181241</u>**
0.30	1	0.423743	2	**<u>0.4181241</u>**
0.35	1	0.423743	5	0.4499825
0.40	1	0.423743	5	0.4499825
0.45	1	0.423743	5	0.4499825
0.50	1	0.423743	13	0.4999935
0.55	1	0.423743	8	0.4359098
0.60	1	0.423743	8	0.4359098
0.65	1	0.423743	8	0.4978527
0.70	1	0.423743	7	0.4783967
0.75	1	0.423743	7	0.481209
0.80	1	0.423743	6	0.4822992
0.85	1	0.423743	6	0.4961151
0.90	1	0.423743	6	0.4961151
0.95	1	0.423743	4	0.4545432
1.00	1	0.274255	1	0.4237425

Table 9 shows the results from HOEP comparing with ones from k-means and k-medoids algorithms. The number of clusters from the result of HOEP algorithm is set as the predetermined value in the initial step of k-means and k-medoids algorithms. Therefore, we compare the total variance of the same number of clusters in each algorithm. The results from normalized function $g_1(f_i)$ show that the total variance from HOEP is lower than the total variance from both of k-means algorithm and k-medoids algorithm. Considering normalized function $g_2(f_i)$, the total variances from HOEP are higher than total variances from k-means and k-medoids algorithms. This means that the normalized function $g_1(f_i)$ may be more suitable for seed dataset than the normalized function $g_2(f_i)$. However, if we do not consider the gamma value that gives the minimum total variance in HOEP, the results of the total variances from HOEP may be lower than total variances from k-means and k-medoids algorithms. (See Table 9)

Table 9 Comparison of the total variance of seed dataset

f_i^*	γ	The number of clusters	Total variance		
			HOEP	k-means	k-medoids
$g_1(f_i)$	0.10	4	0.3425730	0.3555230	0.3434717
$g_2(f_i)$	0.10 0.15 0.20 0.25 0.30	2	0.4181241	0.3209425	0.3290560
	0.55	8	0.4359098	0.4694328	0.4730751
	0.60	8	0.4359098	0.4693166	0.4724625

E.coli dataset

Table 10 shows the number of clusters and the total variance of the E.coli dataset from HOEP algorithm. The values of gamma are varied from 0.05 to 1 with 0.05 increments. The result shows that the gamma values that give the minimum total variance in HOEP algorithm for normalized function $g_1(f_i)$ are 0.05, 0.90, 0.95 and 1.0 and for normalized function $g_2(f_i)$ are 0.05, 0.10, 0.15, 0.20 and 1.00.

Table 10 Results of E.coli dataset from HOEP algorithm

	$f_i^* = g_1(f_i) = \frac{f_i}{\lvert S \rvert}$		$f_i^* = g_2(f_i) = \frac{f_i - \min_j f_j}{\max_j f_j - \min_j f_j}$	
γ	The number of clusters	Total variance HOEP	The number of clusters	Total variance HOEP
0.05	1	<u>**0.223759**</u>	1	<u>**0.223759**</u>
0.10	2	0.241686	1	<u>**0.223759**</u>
0.15	6	0.365499	1	<u>**0.223759**</u>
0.20	5	0.377571	1	<u>**0.223759**</u>
0.25	7	0.401987	2	0.2416859
0.30	5	0.52312	2	0.2416859
0.35	4	0.510741	5	0.3362595
0.40	2	0.403802	9	0.4770188
0.45	2	0.403802	8	0.6150343
0.50	2	0.403802	5	0.3775712
0.55	2	0.403802	3	0.3564157
0.60	2	0.403802	3	0.3564157
0.65	2	0.403802	3	0.3564157
0.70	2	0.403802	3	0.3564157
0.75	2	0.403802	5	0.3853661
0.80	2	0.403802	14	0.7245408
0.85	2	0.403802	14	0.7245408
0.90	1	<u>**0.223759**</u>	12	0.6833156
0.95	1	<u>**0.223759**</u>	11	0.6846903
1.00	1	<u>**0.223759**</u>	1	<u>**0.223759**</u>

From the above table, the calculated gamma values give the same number of clusters which is only one cluster. This means that the values of total variance from three algorithms are equal. However, when we choose other gamma values which are not the gamma values that give the minimum total variance in HOEP, the results from HOEP are better than the compared algorithms as seen in Table 11.

In Table 11, from the normalized function $g_1(f_i)$, we vary the values of gamma varied from 0.05 to 0.25 with 0.05 incremental. The results show that the total variance from HOEP is lower than the total variances from k-means and k-medoids algorithms. This is the same result when we set the gamma values are equal to 0.25, 0.30 and 0.35 using the normalized function $g_2(f_i)$.

Table 11 Comparison of the total variance of E.coli dataset

f_i^*	γ	The number of clusters	Total variance		
			HOEP	k-means	k-medoids
$g_1(f_i)$	**0.05**	**1**	**0.223759**	**0.223759**	**0.223759**
	0.10	2	0.241686	0.298999	0.291798
	0.15	6	0.365499	0.501051	0.501281
	0.20	5	0.377571	0.457003	0.474829
	0.25	7	0.401987	0.538358	0.525647
$g_2(f_i)$	**0.05** **0.10** **0.15** **0.20**	**1**	**0.223759**	**0.223759**	**0.223759**
	0.25	2	0.241686	0.301083	0.291798
	0.30	2	0.241686	0.298107	0.291798
	0.35	5	0.336260	0.460839	0.462371
	1.00	**1**	**0.223759**	**0.223759**	**0.223759**

Wine dataset

Table 12 demonstrates the number of clusters and the total variance of the wine dataset from HOEP algorithm. The values of gamma are varied from 0.05 to 1 with 0.05 increments. The result shows that the gamma values that give the minimum total variance in HOEP algorithm for normalized function $g_1(f_i)$ are 0.40, 0.45, 0.50, ..., 1.0 and for normalized function $g_2(f_i)$ are 0.05 and 0.10.

Table 12 Results of wine dataset from HOEP algorithm

	$f_i^* = g_1(f_i) = \frac{f_i}{\lvert S \rvert}$		$f_i^* = g_2(f_i) = \frac{f_i - \min_j f_j}{\max_j f_j - \min_j f_j}$	
γ	The number of clusters	Total variance HOEP	The number of clusters	Total variance HOEP
0.05	3	0.826678	1	**<u>0.5370761</u>**
0.10	4	0.917629	1	**<u>0.5370761</u>**
0.15	4	1.010106	3	0.8266777
0.20	8	1.621768	4	0.9176289
0.25	8	1.790671	4	0.9176289
0.30	3	1.152957	4	0.9176289
0.35	2	0.815026	4	0.9176289
0.40	1	**<u>0.537076</u>**	4	1.0101061
0.45	1	**<u>0.537076</u>**	4	1.0101061
0.50	1	**<u>0.537076</u>**	10	1.8864756
0.55	1	**<u>0.537076</u>**	10	1.9952272
0.60	1	**<u>0.537076</u>**	10	1.9952272
0.65	1	**<u>0.537076</u>**	9	1.9096509
0.70	1	**<u>0.537076</u>**	8	1.8918815
0.75	1	**<u>0.537076</u>**	8	1.8918815
0.80	1	**<u>0.537076</u>**	8	1.7906712
0.85	1	**<u>0.537076</u>**	7	1.9546810
0.90	1	**<u>0.537076</u>**	6	1.8737017
0.95	1	**<u>0.537076</u>**	6	1.8737017
1.00	1	**<u>0.537076</u>**	1	0.5370761

In Table 12, the provided gamma values give the same number of clusters which is only one cluster in dataset. This means that the values of total variance from three algorithms are equal because there is no partitioning of instances into clusters. However, when we choose other gamma values which are not the gamma values that give the minimum total variance in HOEP, the results from HOEP are better than both k-means and k-medoids algorithms.

In Table 13, from the normalized function $g_1(f_i)$, we vary the values of gamma from 0.05 to 0.20 with 0.05 incremental. The results show that the total variance from HOEP is lower than the total variances from k-means and k-medoids algorithms except at the gamma value equal to 0.05 and 0.15 the total variance from k-means is lower than total variance from HOEP.

From the normalized function $g_2(f_i)$, the values of gamma are varied from 0.10 to 0.50 with 0.05 incremental. The results show that the total variances from HOEP are lower than the total variances from k-means and k-medoids algorithms except at the gamma value equal to 0.15 the total variance from k-means is lower than total variance from HOEP.

Table 13 The comparison of the total variance of wine dataset

f_i^*	γ	The number of clusters	Total variance		
			HOEP	k-means	k-medoids
$g_1(f_i)$	0.05	3	0.826678	0.826381	0.8378864
	0.10	4	0.917629	1.03006	1.0201929
	0.15	4	1.010106	1.037033	1.0177082
	0.20	8	1.621768	1.686746	1.6869938
	0.40	**1**	**0.537076**	**0.537076**	**0.537076**
$g_2(f_i)$	**0.05**	**1**	**0.537076**	**0.537076**	**0.537076**
	0.10	**1**	**0.537076**	**0.537076**	**0.537076**
	0.15	3	0.8266777	0.823341	0.8378864
	0.20	4	0.9176289	1.0380546	1.0224664
	0.25	4	0.9176289	1.0274516	1.0190927
	0.30	4	0.9176289	1.0364914	1.0274369
	0.35	4	0.9176289	1.0377276	1.0167909
	0.40	4	1.0101061	1.0388452	1.0193632
	0.45	4	1.0101061	1.0323878	1.0245189
	0.50	10	1.8864756	1.9247291	1.9322233

CHAPTER V
CONCLUSION

The Half-orbital extreme pole algorithm is a clustering algorithm which uses a concept of the farthest pair to partition instances into groups. The concept of the farthest pair is derived from two papers in classification [14, 15]. In our experiment, we compare our algorithm with k-means clustering algorithm and k-medoids clustering algorithm using 4 datasets which are iris, E.coli, seed, and wine datasets. The performance measure used in this thesis is total variance. We use an average value of total variances from 100 rounds of experiments to compare the performance of all three algorithms with respect to the same number of clusters.

Even though HOEP algorithm does not require the number of clusters in the beginning, HOEP algorithm requires the value of parameter gamma. We suggest the method to select a gamma value by the following technique. First, gamma values are varied from 0.05 to 1 with 0.05 increments. Then, in every run, the total variances are calculated and collected. Finally, the suggested gamma is the gamma value that gives the minimum total variance.

Using the gamma selecting technique, the results of the Iris dataset show that both transformed functions of our algorithm give the same three final clusters whose the total variance is lower than those from both k-means and k-medoids algorithms.

For the results of the Seed dataset, the suggested gamma value is good for the first transformed function because it provides the final clusters that their total variance is lower than total variances from k-means and k-medoids algorithm. For the second transformed function, the total variance from HOEP algorithm with respect to the suggested gamma value is higher than those from k-means and k-medoids algorithms.

In the E.coli and the Wine datasets, a suggested gamma value gives only one final cluster. Since the cluster formed by all instances has the total variance lower than the total variance from any partitioned clusters.

In conclusion, the HOEP algorithm tries to form instances into clusters by starting from boundary. The visualization from Principal Component Analysis (PCA) cannot demonstrate this fact in some datasets because PCA tries to visualize a dataset which has several attributes into two-dimensional space. However, the scatter plots of two attributes can show this fact. HOEP algorithm can perform well when the appropriate gamma value is chosen. The appropriate gamma value provides the final clusters that have low total variance.

In the first iteration, if HOEP algorithm is terminated by the criterion that there is only one instance which is partitioned, the outlier is first detected. As the result of this criterion, HOEP algorithm is good for a dataset that contains outliers. Furthermore, this algorithm can be used to detect the outliers of a dataset by considering the number of instances from the cluster that is formed first.

Furthermore, HEOP algorithm is suitable for a dataset whose constructed histogram is nonhomogeneous because if the appropriate gamma value is chosen, the instances can be grouped into the appropriate clusters. Since each bin of histogram is constructed from the frequency of instances in each layer, the nonhomogeneous histogram is occurred when there is the middle region of layers of ball that has the density of instances lower than the density of instances of its left and its right regions of layers of ball.

HOEP algorithm has three major advantages. The first advantage is that the number of clusters is not needed to be set by users. The second advantage is that the distances between every instance and centroid are not needed to be re-calculated after the first distance calculation. The other advantage is the finalized clustering will be the same for different runs. These can rectify three drawbacks of k-means and k-medoids algorithms which are the number of clusters is required to be predetermined, the distances between instances and their centroids are needed to be recalculated in every iteration and the different initial centroids can result in different final clusters.

However, the HOEP algorithm cannot partition a cluster from a dataset that the constructed histogram is homogeneous because the gamma value cannot find the splitting point. Thus, our algorithm can be improved by using other methods to split instances into clusters instead of using histogram. Furthermore, our algorithm can also be improved by finding an appropriate way to determine the parameter gamma or using different values of gamma in each iteration, because each iteration might require its own gamma values to perform effectively.

REFERENCES

[1] Quinlan, J. R. (1993). C4.5: programs for machine learning, Morgan kaufmann.

[2] Cover, T. and P. Hart (1967). "Nearest neighbor pattern classification." Information Theory, IEEE Transactions on **13**(1): 21-27.

[3] John, G. H. and P. Langley (1995). "Estimating continuous distributions in Bayesian classifiers." Proceedings of the Eleventh conference on Uncertainty in artificial intelligence, Morgan Kaufmann Publishers Inc.

[4] Hearst, M. A., et al. (1998). "Support vector machines." Intelligent Systems and their Applications, IEEE Transactions on **13**(4): 18-28.

[5] Wold, S., et al. (1987). "Principal component analysis." Chemometrics and intelligent laboratory systems **2**(1): 37-52.

[6] Kohonen, T. (1990). "The self-organizing map." Proceedings of the IEEE **78**(9): 1464-1480.

[7] MacQueen, J. (1967). "Some methods for classification and analysis of multivariate observations." Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, California, USA.

[8] Hartigan, J. A. and M. A. Wong (1979). "Algorithm AS 136: A k-means clustering algorithm." Journal of the Royal Statistical Society. Series C (Applied Statistics) **28**(1): 100-108.

[9] Lloyd, S. (1982). "Least squares quantization in PCM." Information Theory, IEEE Transactions on **28**(2): 129-137.

[10] Kaufman, L. and P. Rousseeuw (1987). "Clustering by means of medoids." in Statistical Data Analysis Based on the L_1–Norm and Related Methods, edited by Y. Dodge, North-Holland, 405–416

[11] Kaufman, L. R. and P. Rousseeuw (1990). Finding groups in data: An introduction to cluster analysis, Wiley Series in Probability and Mathematical Statistics. Applied Probability and Statistics, New York.

[12] Ng, R. T. and J. Han (1994). "Efficient and Effective Clustering Methods for Spatial Data Mining. " Proceedings of the twevelth conference on Very Large Data Bases (VLDB), 144-155.

[13] Park, H.-S. and C.-H. Jun (2009). "A simple and fast algorithm for K-medoids clustering." Expert Systems with Applications **36**(2): 3336-3341.

[14] Sinapiromsaran, K. and N. Techaval (2012). "Network intrusion detection using multi-attributed frame decision tree." Proceedings of the Second conference on Digital Information and Communication Technology and it's Applications (DICTAP), 2012 the 10th International Conference on IEEE, 203-207

[15] Sirisomboonrat, C. and K. Sinapiromsaran (2012). "Breast cancer diagnosis using multi-attributed lens recursive partitioning algorithm. " Proceedings of the tenth conference on ICT and Knowledge Engineering (ICT & Knowledge Engineering), 2012 the 10th International Conference on IEEE, 40-45

[16] Kanungo, T., et al. (2002). "An efficient k-means clustering algorithm: analysis and implementation." Pattern Analysis and Machine Intelligence, IEEE Transactions on **24**(7): 881-892.

[17] Sturges, H. A. (1926). "The choice of a class interval." Journal of the American Statistical Association **21**(153): 65-66.

[18] Scott, D. W. (1992). Multivariate Density Estimation: Theory, Practice, and Visualization. John Wiley and Sons, New York: 47-48.

APPENDIX

APPENDIX : RESULTS FROM HALF-ORBITAL ALGORITHM

Iris dataset

Table 14 The results from iris dataset

	$f_i^* = g_1(f_i) = \frac{f_i}{\lvert S \rvert}$				$f_i^* = g_2(f_i) = \frac{f_i - \min_j f_j}{\max_j f_j - \min_j f_j}$			
γ	k	Total variance			k	Total variance		
		HOEP	k-means	k-medoid		HOEP	k-means	k-medoid
0.05	1	0.274255	0.274255	0.2742545	1	0.2742545	0.2742545	0.2742545
0.10	3	0.139462	0.147885	0.1399325	1	0.2742545	0.2742545	0.2742545
0.15	2	0.270116	0.139887	0.1398865	1	0.2742545	0.2742545	0.2742545
0.20	5	0.257964	0.153486	0.1530062	5	0.1485446	0.1541574	0.1519457
0.25	3	**0.137439**	0.147823	0.1398882	5	0.1485446	0.1526397	0.1519537
0.30	2	0.139887	0.139887	0.1398865	6	0.2736472	0.1623842	0.1623433
0.35	1	0.274255	0.274255	0.2742545	3	**0.1394619**	0.1496138	0.1398587
0.40	1	0.274255	0.274255	0.2742545	3	**0.1394619**	0.1466788	0.1399226
0.45	1	0.274255	0.274255	0.2742545	3	0.1433193	0.147432	0.1399522
0.50	1	0.274255	0.274255	0.2742545	4	0.1440115	0.147392	0.1490768
0.55	1	0.274255	0.274255	0.2742545	4	0.1440115	0.148028	0.1492465
0.60	1	0.274255	0.274255	0.2742545	4	0.1440115	0.1471689	0.1481208
0.65	1	0.274255	0.274255	0.2742545	10	0.1926997	0.1823195	0.1889037
0.70	1	0.274255	0.274255	0.2742545	2	0.2701162	0.1398865	0.1398865
0.75	1	0.274255	0.274255	0.2742545	6	0.248374	0.1623894	0.1631725
0.80	1	0.274255	0.274255	0.2742545	6	0.248374	0.1620667	0.1624493
0.85	1	0.274255	0.274255	0.2742545	6	0.248374	0.1612586	0.1621937
0.90	1	0.274255	0.274255	0.2742545	4	0.2418484	0.1461825	0.1486243
0.95	1	0.274255	0.274255	0.2742545	5	0.2450899	0.1553587	0.1519796
1.00	1	0.274255	0.274255	0.2742545	1	0.2742545	0.2742545	0.2742545

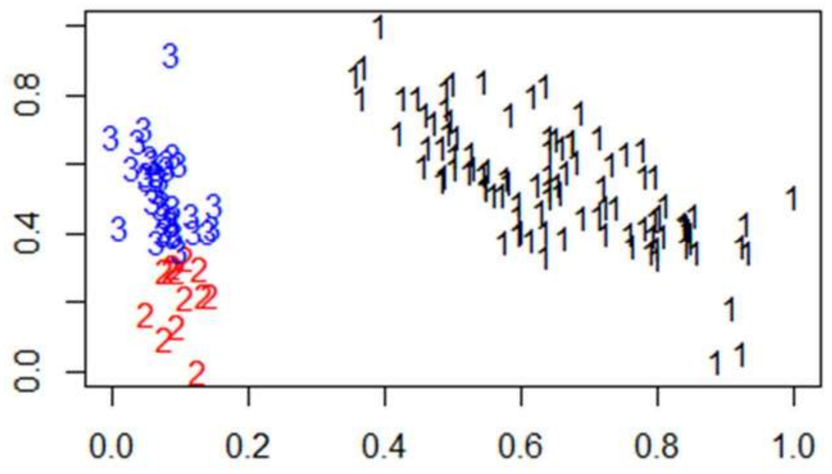


Figure 15 Result of iris dataset from HOEP algorithm with the first transformed function and the gamma value as 0.25

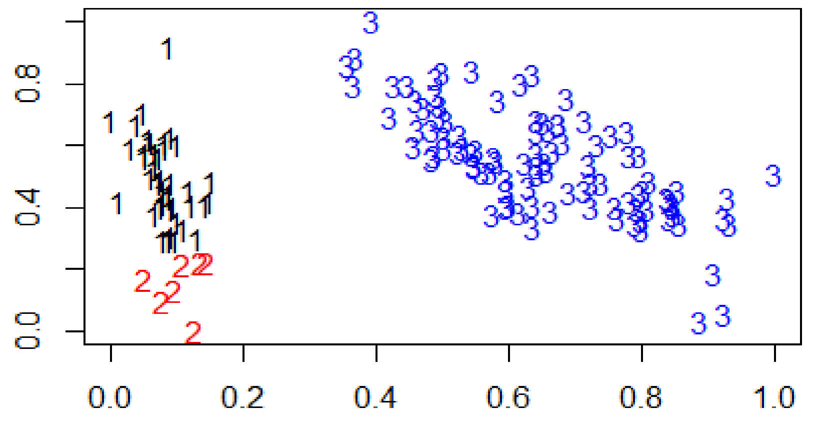


Figure 16 Result of iris dataset from HOEP algorithm with the second transformed function and the gamma value as 0.35

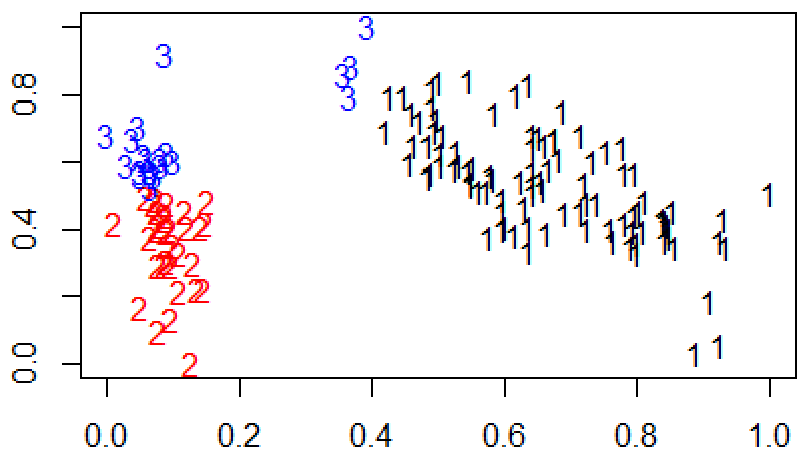


Figure 17 Result of iris dataset from k-means algorithm (k = 3)

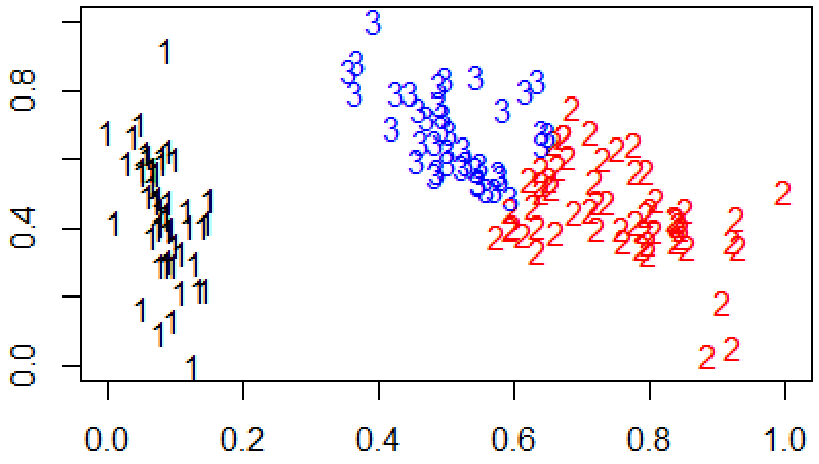


Figure 18 Result of iris dataset from k-medoids algorithm (k = 3)

Seed dataset

Table 15 The results from seed dataset

	$f_i^* = g_1(f_i) = \frac{f_i}{\lvert S\rvert}$				$f_i^* = g_2(f_i) = \frac{f_i - \min_j f_j}{\max_j f_j - \min_j f_j}$			
γ	k	Total variance			k	Total variance		
		HOEP	k-means	k-medoid		HOEP	k-means	k-medoid
0.05	2	0.418124	0.320801	0.3288243	1	0.4237425	0.4237425	0.4237425
0.10	4	**0.342573**	0.355523	0.3434717	2	**0.4181241**	0.3209425	0.3281293
0.15	3	0.386164	0.3151	0.3159005	2	**0.4181241**	0.3208924	0.3283609
0.20	1	0.423743	0.423743	0.4237425	2	**0.4181241**	0.3207223	0.329056
0.25	1	0.423743	0.423743	0.4237425	2	**0.4181241**	0.3209959	0.3258125
0.30	1	0.423743	0.423743	0.4237425	2	**0.4181241**	0.3209675	0.3272025
0.35	1	0.423743	0.423743	0.4237425	5	0.4499825	0.3874692	0.3739612
0.40	1	0.423743	0.423743	0.4237425	5	0.4499825	0.388273	0.3733803
0.45	1	0.423743	0.423743	0.4237425	5	0.4499825	0.3892873	0.3738341
0.50	1	0.423743	0.423743	0.4237425	13	0.4999935	0.5606782	0.5821383
0.55	1	0.423743	0.423743	0.4237425	8	0.4359098	0.4694328	0.4730751
0.60	1	0.423743	0.423743	0.4237425	8	0.4359098	0.4693166	0.4724625
0.65	1	0.423743	0.423743	0.4237425	8	0.4978527	0.4710116	0.4703925
0.70	1	0.423743	0.423743	0.4237425	7	0.4783967	0.4477825	0.4419135
0.75	1	0.423743	0.423743	0.4237425	7	0.481209	0.447614	0.4415455
0.80	1	0.423743	0.423743	0.4237425	6	0.4822992	0.4182504	0.4093491
0.85	1	0.423743	0.423743	0.4237425	6	0.4961151	0.4189727	0.4101843
0.90	1	0.423743	0.423743	0.4237425	6	0.4961151	0.417759	0.4077891
0.95	1	0.423743	0.423743	0.4237425	4	0.4545432	0.3552202	0.3435858
1.00	1	0.423743	0.423743	0.4237425	1	0.4237425	0.4237425	0.4237425

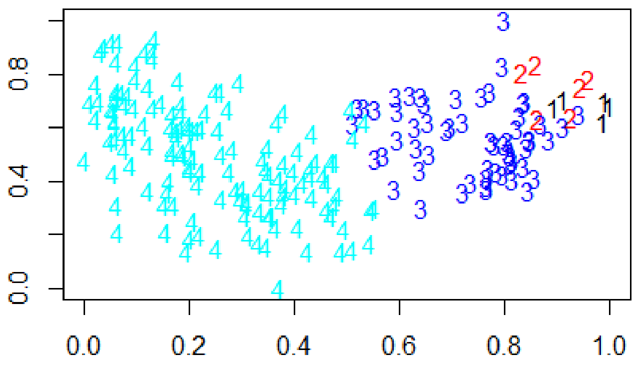


Figure 19 Result of seed dataset from HOEP algorithm with the first transformed function and the gamma value as 0.10

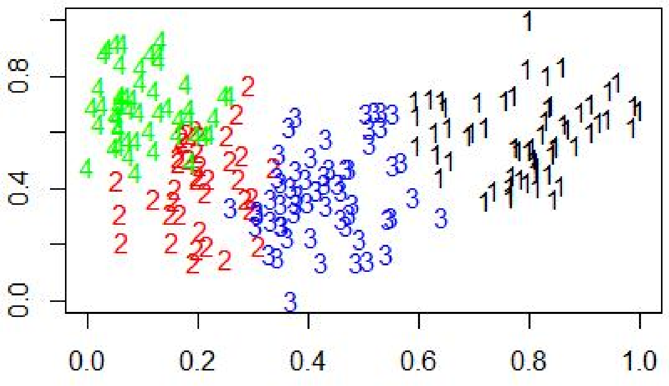


Figure 20 Result of seed dataset from k-means algorithm (k = 4)

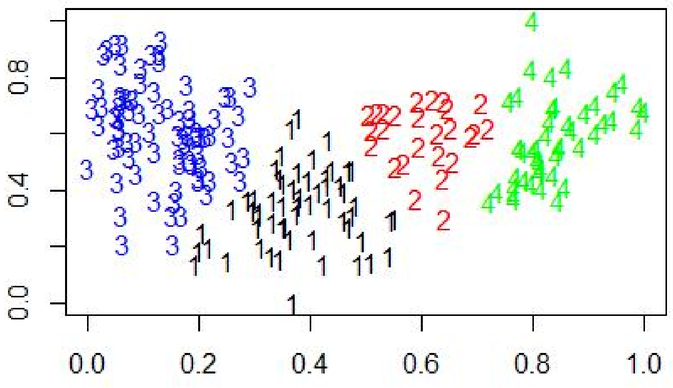


Figure 21 Result of seed dataset from k-medoids algorithm (k = 4)

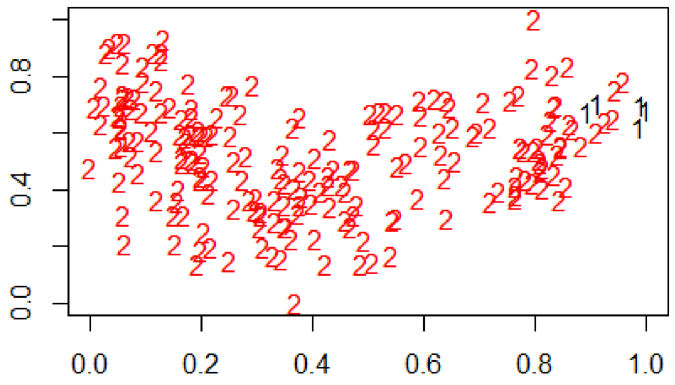


Figure 22 Result of seed dataset from HOEP algorithm with the second transformed function and the gamma value as 0.1

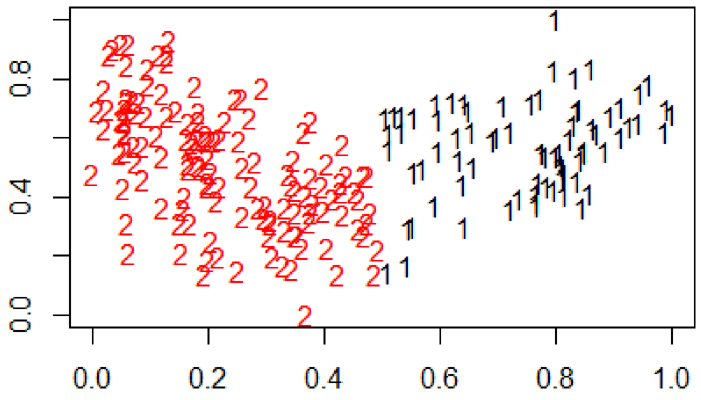


Figure 23 Result of seed dataset from k-means algorithm (k = 2)

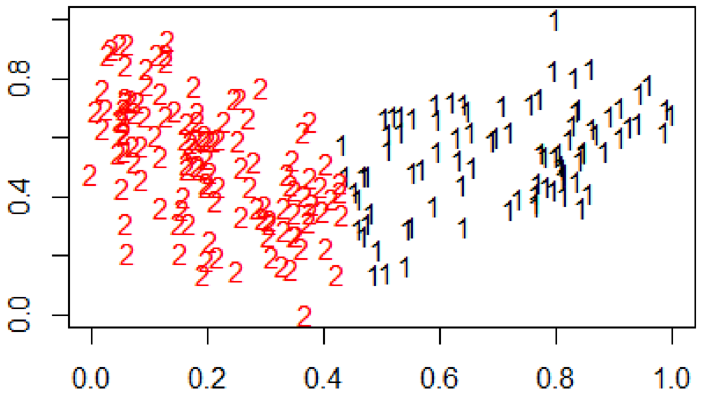


Figure 24 Result of seed dataset from k-medoids algorithm (k = 2)

E.coli Dataset

Table 16 The results from E.coli dataset

	$f_i^* = g_1(f_i) = \frac{f_i}{\|S\|}$				$f_i^* = g_2(f_i) = \frac{f_i - \min_j f_j}{\max_j f_j - \min_j f_j}$		
γ	k	Total variance			k	Total variance		
		HOEP	k-means	k-medoid		HOEP	k-means	k-medoid
0.05	1	**0.223759**	0.223759	0.223759	1	**0.223759**	0.223759	0.223759
0.10	2	0.241686	0.298999	0.2917981	1	**0.223759**	0.223759	0.223759
0.15	6	0.365499	0.501051	0.5012805	1	**0.223759**	0.223759	0.223759
0.20	5	0.377571	0.457003	0.4748292	1	**0.223759**	0.223759	0.223759
0.25	7	0.401987	0.538358	0.5256469	2	0.2416859	0.2943786	0.2917981
0.30	5	0.52312	0.460468	0.4736863	2	0.2416859	0.2897843	0.2917981
0.35	4	0.510741	0.410694	0.4280872	5	0.3362595	0.4627097	0.4743868
0.40	2	0.403802	0.300095	0.2917981	9	0.4770188	0.596143	0.575553
0.45	2	0.403802	0.29122	0.2917981	8	0.6150343	0.5543817	0.5481004
0.50	2	0.403802	0.299561	0.2917981	5	0.3775712	0.4590082	0.4753218
0.55	2	0.403802	0.294654	0.2917981	3	0.3564157	0.3522067	0.3452243
0.60	2	0.403802	0.304923	0.2917981	3	0.3564157	0.3531643	0.3452243
0.65	2	0.403802	0.292763	0.2917981	3	0.3564157	0.3562084	0.3452243
0.70	2	0.403802	0.307255	0.2917981	3	0.3564157	0.3567013	0.3452243
0.75	2	0.403802	0.300702	0.2917981	5	0.3853661	0.4617241	0.4717096
0.80	2	0.403802	0.2933	0.2917981	14	0.7245408	NaN*	0.6959607
0.85	2	0.403802	0.295378	0.2917981	14	0.7245408	NaN*	0.6971183
0.90	1	**0.223759**	0.223759	0.223759	12	0.6833156	0.6636169	0.6512241
0.95	1	**0.223759**	0.223759	0.223759	11	0.6846903	0.6453578	0.6294429
1.00	1	**0.223759**	0.223759	0.223759	1	**0.223759**	0.223759	0.223759

*NaN in k-means algorithm means that the k-means algorithm does not converge to the final clusters after 50 iterations.

<u>Wine dataset</u>

Table 17 The results from wine dataset

	$f_i^* = g_1(f_i) = \frac{f_i}{\lvert S\rvert}$				$f_i^* = g_2(f_i) = \frac{f_i - \min_j f_j}{\max_j f_j - \min_j f_j}$			
γ	k	Total variance			k	Total variance		
		HOEP	k-means	k-medoid		HOEP	k-means	k-medoid
0.05	3	0.826678	0.826381	0.8378864	1	**<u>0.5370761</u>**	0.5370761	0.5370761
0.10	4	0.917629	1.03006	1.0201929	1	**<u>0.5370761</u>**	0.5370761	0.5370761
0.15	4	1.010106	1.037033	1.0177082	3	0.8266777	0.823341	0.8378864
0.20	8	1.621768	1.686746	1.6869938	4	0.9176289	1.0380546	1.0224664
0.25	8	1.790671	1.676395	1.6850448	4	0.9176289	1.0274516	1.0190927
0.30	3	1.152957	0.828306	0.8378864	4	0.9176289	1.0364914	1.0274369
0.35	2	0.815026	0.731775	0.7256739	4	0.9176289	1.0377276	1.0167909
0.40	1	**<u>0.537076</u>**	0.537076	0.5370761	4	1.0101061	1.0388452	1.0193632
0.45	1	**<u>0.537076</u>**	0.537076	0.5370761	4	1.0101061	1.0323878	1.0245189
0.50	1	**<u>0.537076</u>**	0.537076	0.5370761	10	1.8864756	1.9247291	1.9322233
0.55	1	**<u>0.537076</u>**	0.537076	0.5370761	10	1.9952272	1.9179555	1.9264655
0.60	1	**<u>0.537076</u>**	0.537076	0.5370761	10	1.9952272	1.9314268	1.9185647
0.65	1	**<u>0.537076</u>**	0.537076	0.5370761	9	1.9096509	1.7987976	1.787649
0.70	1	**<u>0.537076</u>**	0.537076	0.5370761	8	1.8918815	1.6660726	1.6764645
0.75	1	**<u>0.537076</u>**	0.537076	0.5370761	8	1.8918815	1.6854133	1.6781531
0.80	1	**<u>0.537076</u>**	0.537076	0.5370761	8	1.7906712	1.6856917	1.6846281
0.85	1	**<u>0.537076</u>**	0.537076	0.5370761	7	1.954681	1.5438785	1.5625041
0.90	1	**<u>0.537076</u>**	0.537076	0.5370761	6	1.8737017	1.3795635	1.3819696
0.95	1	**<u>0.537076</u>**	0.537076	0.5370761	6	1.8737017	1.3867941	1.3842947
1.00	1	**<u>0.537076</u>**	0.537076	0.5370761	1	0.5370761	0.5370761	0.5370761

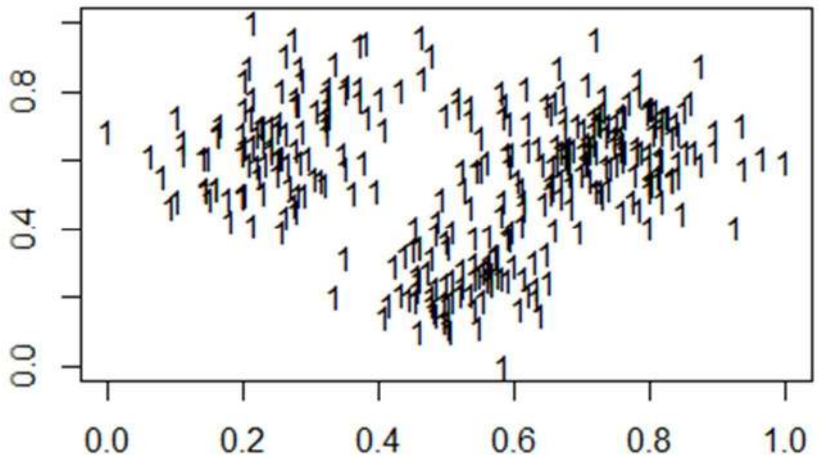


Figure 25 Results of E.coli dataset from three algorithms are the same.

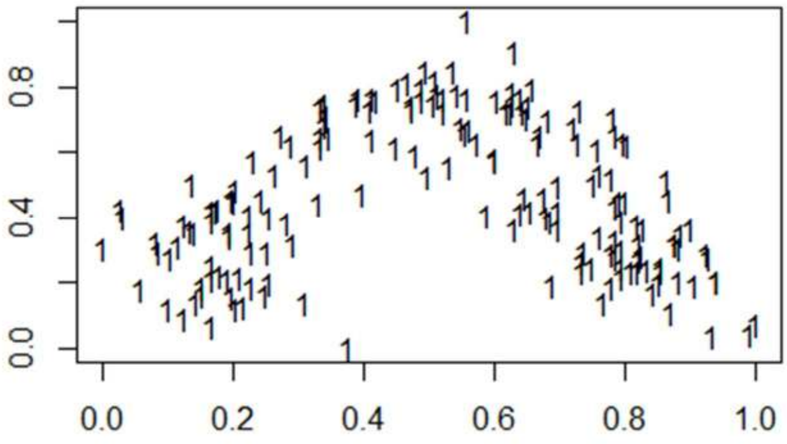


Figure 26 Results of wine dataset from three algorithms are the same.

VITA

Name	Benjapun Kaveelerdpotjana
Date of Birth	3 October 1989
Place of Birth	Bangkok, Thailand
Education	B.Sc.(Mathematics), Kasetsart University, 2010
Scholarship	The Development and Promotion of Science and Technology Talents project (DPST)
Publication	Benjapun Kaveelerdpotjana, Krung Sinapiromsaran, Boonyarit Intiyot, Farthest Boundary Clustering Algorithm: Half-orbital Extreme Pole, The seventeenth International Computer Science and Engineering Conference (ICSEC) (2013): 173-178