CENTRE-BASED HARD CLUSTERING ALGORITHMS FOR Y-STR DATA

Ali Seman, **Zainab Abu Bakar**, **Azizian Mohd. Sapawi**

Department of Computer Sciences,
Faculty of Computer and Mathematical Sciences,
Universiti Teknologi MARA (UiTM)
40450 Shah Alam, Selangor
{aliseman; zainab; azizian}@tmsk.uitm.edu.my

Abstract. This paper presents Centre-based hard clustering approaches for clustering Y-STR data. Two classical partitioning techniques: Centroid-based partitioning technique and Representative object-based partitioning technique are evaluated. The k-Means and the k-Modes algorithms are the fundamental algorithms for the centroid-based partitioning technique, whereas the k-Medoids is a representative object-based partitioning technique. The three algorithms above are experimented and evaluated in partitioning Y-STR haplogroups and Y-STR Surname data. The overall results show that the centroid-based partitioning technique is better than the representative object-based partitioning technique in clustering Y-STR data.

Keywords: *Centre-based clustering, k-Means, k-Modes, k-Medoids, Y-STR data*

1. Introduction

Centre-based clustering algorithms are very efficient especially for clustering large databases and high-dimensional databases (Gan et al. 2007). The pillar of the Centre-based clustering algorithm is *k*-Means clustering algorithm introduced at almost three decade ago by Macqueen (1967). The *k*-Means paradigm depends an initial *k*, in which known as a priori, utilizes means as a mechanism to update centroids and normally opts Euclidean distance as the dissimilarity measure. As consequent, the *k*-Means paradigm has been extended significantly regardless the types of data. For examples, *k*-Modes algorithm proposed by Huang (1998) that handled specifically categorical data used also *k*-Means paradigm. Huang (1998) also introduced *k*-Prototypes algorithm that combined *k*-Means and *k*-Modes algorithms for mixed data types. The paradigm is also extended by Kaufman & Rousseeuw (1987) when they introduced the idea of *k*-Mediods algorithm or Partitioning Around Medioids (PAM).

Further, a lot of extended *k*-Means paradigm have been introduced such as Continuous k-Means algorithm (Faber, 1994), Compare-means algorithm (Philips, 2002), fuzzy covariance clustering (Gustafson and Kessel, 1979) and Fuzzy c-Elliptotypes algorithm (Bezdek, 1981). This includes also the variation of *k*-Modes algorithms such as *k*-Modes with new dissimilarity measures by He et al. (2007) and Ng et al. (2007), *k*-Population (Kim et al. 2007), a new fuzzy k-Modes proposed by Ng & Jing (2009). For *k*-Medoids, the main extended versions are Clustering LARGE Applications (CLARA) by Kaufman & Rousseeuw (1990) and Clustering Large Applications based upon Randomized Search (CLARANS) by Ng & Han (1994).

ISSN 2231-7473

However, in clustering approaches, there is no effort has been observed clustering Y-STR data except recently there is one in Centre-based clustering (Ali et al. 2010). The results show that the clustering methods can be used in Y-STR data, in fact the data can be treated in both data types: numerical and categorical data. Thus, the aim of this paper is to investigate the clustering performance based on: (1) Centroid-based partitioning technique and (2) Representative Object-based partitioning technique. For the centroid-based partitioning technique, the focus is to investigate the classical hard *k*-Means by Macqueen (1967) for numerical Y-STR data and hard *k*-Modes by Huang (1998) algorithms for Y-STR categorical data only. Consequently, the *k*-Medoids by Kaufman and Rousseeuw (1987) is chosen for the second technique. The objective of the investigation is to fundamentally evaluate the partitioning techniques applied for Y-STR data and its performances.

2. Y-STR data and Its Applications

Y-STR is Short Tandem Repeats on Y-Chromosome. The Y-STR data represents the number of times an STR repeats, called allele value for each marker. If a Y-STR marker, say DYS391, the tandem repeats are: [TCTA] [TCTA] [TCTA] [TCTA] [TCTA] [TCTA] [TCTA] [TCTA], thus the allele value is counted as eight. This DNA method is now actively used in Anthropological Genetics as well as in Genetic Genealogy. Further, this method is a very promising method to support a traditional approach especially in studying human migration patterns and proving genealogical relationships. For further information, the Y-STR used in Anthropology can be found in a book called Anthropological Genetics: Theory, Methods and Apllications (2007) and for Genetic Genealogy can be found in Fitzpatrick (2005), and Fitzpatrick & Yeiser(2005).

The genetic distance for a person may differ from other by referring the allele values for each marker. If a person shares the same allele value for each marker is considered coming from the same ancestor from genealogical perspective. In a broader perspective, for instance in studying human migration patterns, it can be under the same haplogroups which includes different geographical area throughout the world. The Y-STR data can be grouped into meaningful groups based on the distance for each STR marker. For genealogical data such as Y-Surname project, the distances are based on 0 or 1 or 2 or 3 mismatches, whereas the haplogroups are determined by a method known as Single Nucleotide Polymorphism (SNP) analysis. There are set of very broad haplogroups and all males in the world can be placed into a system of defining Y-DNA haplogroups by letters A through to T, with further subdivisions using numbers and lower case letters. See International Society of Genetic Genealogy (www.isogg.org). The haplogroups have been established by the Y Chromosome Consortium (YCC). For further details, see University of Arizona (http://ycc.biosci.arizona.edu/).

3. Notation

Let $X = \{X_1, X_2, \ldots, X_n\}$ be set of *n* Y-STR data and $A = \{A_1, A_2, \ldots, A_m\}$ be set of markers/attributes of Y-STR. We define A_j is the *j*th attribute values as associated *j*th marker with the actual STR allele value. We define X is a numerical data if it is treated only as numerical values as it is. Note that the Y-STR data are originally a numeric domain as associated with the allele values and it is discrete values. We define X is a categorical data if it is treated only as categorical values. Note that for each attribute A_j describes a domain values, denoted by $DOM(A_j)$. A domain $DOM(A_j)$ is defined as categorical data if it is finite and unordered, e.g., for any $a,b \in DOM(A_j)$, either $a=b$ or $a \neq b$. Consider the *j*th attribute values are: $A_j = \{10, 10, 11, 11, 12, 13, 14\}$, thus the $DOM(A_j)=\{10,11,12,13,14\}$. We consider every individual has exactly attribute STR

allele values. If the value of an attribute A_j is missing, then we denote the attribute value of A_j by a category ϵ which means empty. Let X_i be individual, represented as $[x_{i,1}, x_{i,2},...,x_{i,m}]$. We define $X_i = X_k$, if $x_{i,j} = x_{k,j}$ for $1 \leq j \leq m$, where the relation $X_i = X_k$ does not mean that X_i and X_k are the same individual because there exist the two individuals have equal STR allele values in attributes $A_1,A_2,...,A_m$. In Y-STR, there exist a lot of cases; individuals share the same STR allele values throughout markers but different individuals.

4. Classical Hard Partitioning methods

Let us suppose that the objective is to partition a Y-STR data set, *D* consists of *n* Y-STR objects. A classical hard partitioning method constructs partitions, *k* that is known as a priori. Let $X = \{X_1, X_2, \ldots X_n\}$ be set of Y-STR data with set of numeric or categorical attributes $A = \{A_1,A_2,\ldots, A_m\}$. Thus, to partition the Y-STR data, *X* into *k* is to minimize the cost function as Equation (1).

$$P(W,Z) = \sum_{l=1}^{k}\sum_{i=1}^{n} w_{li}\, d(x_i,z_l) \tag{1}$$

Subject to:

$$\sum_{l=1}^{k} w_{li} = 1, \quad 1 \leq i \leq n \tag{2}$$

$$w_{li} \in \{0,1\}, 1 \leq i \leq n, 1 \leq l \leq k \tag{3}$$

and:

$$0 < \sum_{i=1}^{n} w_{li} < n, \quad 1 \leq l \leq k \tag{4}$$

where is a known number of clusters, *W* is a (*k x n*) partition matrix, *Z* = , is the centroids and is a dissimilarity measure between and . Thus, in the case of hard clustering, the object *x* can be assigned into if and only if one cluster based on the nearest objects belong to the *k* partitions as described in Equation (5).

$$w_{li} = \begin{cases} 1, & \text{if } dist(x_i,z_l) = \min dist(x_i,z_l), 1 \leq l \leq k \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

Finally, to achieve global optimality in partition-based clustering, the updating centroids will iteratively be enumerated until the objects stop from moving to the other clusters. Most clustering applications utilize one of two popular heuristic methods (Han & Kamber, 2001): (1) Centroid-based partitioning techniques (CPT) and; (2) Representative Object-based partitioning technique

(ROPT). Thus, the *k*-Means algorithm uses the CPT by calculating the mean values of the objects in the cluster, whereas the *k*-Modes algorithm takes the mode values of the objects in the cluster. However, the *k*-Medoids is a ROPT uses one of the objects located near the centre of the cluster as the medoid.

4.1 The *k*-Means Clustering Algorithm

The *k*-Means algorithm initializes cluster *k* and uses means to calculate the distance between objects and the *k* clusters. The distance measure is normally based on Euclidian distance as in Equation (6). The algorithm allows recalculation of the means of each cluster of the objects belong to it and minimizes the intra cluster dissimilarity. The updating centroid by the means is calculated as in Equation (7).

$$d_{\text{euc}}(x,y) = \left| \sum_{j=1}^{d} (x_j - y_j)^2 \right|^{\frac{1}{2}} \tag{6}$$

$$\text{for } l = 1, 2, \ldots k \text{ and } j = 1, 2, \ldots, d. \tag{7}$$

Figure 1.1 describes the *k*-Means clustering algorithm.

Input: Dataset, *D*, the number of cluster, *k* and the number of dimensional, *d*
Output: A set of clusters, *k*

1: Select *k* initial centroids such that $Z=\{z_1, z_2 \ldots, z_k\}$ and z_l for cluster *l*;
2: repeat
3: **for** *i* = 1 to *n* **do**
4: Find a *l* such that $d_{euc}(x_i,z_l) = \min 1 \leq t \leq k\ d_{euc}(x_i,z_t)$;
5: Allocate x_i to cluster *l*;
6: Recompute the cluster means as in Equation (7) if the clusters changed;
7: **end for**
8: until No objects move
9: Output results

Figure 1.1**:** THE *k*-MEANS CLUSTERING ALGORITHM

4.2 The *k*-Modes Clustering Algorithm

The *k*-Modes algorithm is an extension to the *k*-Means paradigm by focusing on the categorical data, maintaining the initial *k*, replacing the updating centroids by the mode values.

Further, the *k*-Modes algorithm uses a simple matching dissimilarity measure as in Equation (8) and (9).

$$d(X,Y) = \sum_{j=1}^{n} \delta\left(x_j, y_j\right)$$

(8)

where:

$$\delta\left(x_j, y_j\right) = \begin{cases} 0, & x_{ij} = y_{ij} \\ 1, & x_{ij} \neq y_{ij} \end{cases}$$

(9)

The *k*-Modes clustering algorithm utilizes frequency-based method to update modes as in Equation (10) and (11). The algorithm is described in Figure 1.2.

$$z_{lj} = a_j^{(r)}$$

(10)

where is the mode of attribute values of A_j in cluster C_l such that

$$f\left(a_j^{(r)} | C_l\right) \geq f\left(a_j^{(t)} | C_l\right) \forall t, 1 \leq l \leq p_j, a_j^{(r)} \neq a_j^{(t)}$$

(11)

Figure 1.2 describes the *k*-modes clustering algorithm.

Input: Dataset, *D*, the number of cluster, *k* and the number of dimensional, *d*
Output: A set of clusters, *k* **1:** Select *k* initial centroids such that $Z=\{z_1, z_2..., z_k\}$ and z_l for cluster *l*;
2: for *i* = 1 to *n* **do**
3: Find a *l* such that $d_{sim}(x_i,z_l) = \min 1 \leq t \leq k\ d_{sim}(x_i,z_t)$; **4:** Allocate x_i to cluster *l*; **5:** Update the cluster modes as in Equation (10) and (11);
6: end for **7: repeat** **8: for** *i* = 1 to *n* **do**
9: Let l_0 be the index of the cluster of x_i **10:** Find an l_1 such that $d_{sim}(x_i,z_{l1}) = \min d_{sim}(x_i,z_t)$; **11**: if$(d_{sim}(x_i,z_{l1}) < (d_{sim}(x_i,z_{l0})$ then
12: Reallocate x_i to cluster l_1; **13:** Update z_{l0} and z_{l1} as in Equation (10) and (11);

14: end if
15: end for
16: until No objects move
17: Output results

Figure 1.2**:** THE *k*-MODES CLUSTERING ALGORITHM

4.3 The *k*-Medoids Clustering Algorithm

The *k*-Medoids focuses on the objects in which the most centrally located object in a cluster. The basic idea of this algorithm is to find *k* cluster in *n* objects by first arbitrarily finding a representative object, called the medoids for each cluster. The next step is to iteratively replace one of the mediods by one of the non-medoids as long as the process can improve the clustering accuracy. The swapping technique allows exchange the current medoids, t_i and the non-medoids, t_h. The replacement of new medoids must satisfy the total cost, $TC_{ih} < 0$ as in Equation (12).

$$TC_{ih} = \sum_{j=1}^{n} C_{jih} \tag{12}$$

where is the cost change for an item t_j while swapping medoid, t_i with non-medoid, t_h

This algorithm normally uses Euclidean distance as described in Equation (6). The algorithm is described in Figure 1.3.

Input: Dataset, *D*, the number of cluster, *k* and the number of dimensional, *d*
Output: A set of clusters, *k*

1: Select *k* initial objects as the initial clusters;
2: repeat
3: **for** each t_h not a medoid **do**
4: **for** each medoid t_i **do**;
5: calculate TC_{ih} as Equation (12);
6: **end for**
7: Find *i*,*h* where TC_{ih} is the smallest;
8: **if** $TC_{ih} < 0$, **then**
9: replace medoid t_i with t_h;
10: **end if**
11: **end for**
12: until $TC_{ih} > 0$;
13: for each $t_i \in D$ **do**
14: assign t_i to K_j, where $dist(t_i,t_j)$ is the smallest over all mediods;
15: Output results

Figure 1.3**:** THE *k*-MEDOIDS CLUSTERING ALGORITHM

5. Experimental Results

This section discusses on the experimental results for the three algorithms above. Thus, this section explains: (1) Experimental setup and; (3) Clustering performances.

5.1 Experimental Setup

The experiments were conducted on 2 datasets of Y-STR data that were obtained from a database, called worldfamilies.net (www.worldfamilies.net). The first data set is Y-STR data for haplogroup applications. The second data set is Y-STR data for Y-Surname applications. Both data sets are based on 25 markers (attributes). The data sets are as follows:

a) The first data set of Y-STR haplogroup consists of 535 records. The original data were 3419 that consisted of 29 groups. See the complete data in Family Tree DNA (www.familytreedna.com). However, the data had been filtered to chose only 8 groups, called haplogroups, which consist of B(47), D(32), E(12), F(162), H(63), I(123), J(35) and N(61) respectively. The values in the parenthesis indicate the number of records belong to the particular group.

b) The second data set of Y-STR Surname consists of 112 data that belong to Donald Surname. See the details in Donald Surname Project (http://dna-project.clan-donald-usa.org). However, the original of 896 data of Donald Surname had been filtered to obtain only 112 individual based on its modal haplotypes. The modal haplotype for this surname is: 13, 25, 15, 11, 11, 14, 12, 12, 10, 14, 11, 31, 16, 8, 10, 11, 11, 23, 14, 20, 31, 12, 15, 15, 16. Thus, there are 6 classes based on the genetic distance described as mismatches 0 – 5. The mismatches are determined and compared between the individual and its modal haplotypes.

For better results, each dataset and algorithm is runs about 100 times. For each run, the dataset is randomly reordered from the original order. Further, for hard *k*-Means, the distinct initial centroids is chosen to avoid empty clustering, whereas, for hard *k*-Modes, the diverse method is used for initial *k* because the methods had been proved better than the distinct method (see Huang, 1998).

5.2 Clustering Performances

This section discusses on the clustering performances of partitioning Y-STR data regarding the CPT of *k*-Means and *k*-Modes and the ROPT of *k*-Medoids algorithms. Hence, this section presents the experimental results of: (1) clustering accuracy; (2) precision and recall and; (3) time efficiency. Further, for each clustering accuracy, precision and recall, the detail values of average, minimum, maximum and standard deviation are given.

In order to evaluate the clustering accuracy, the misclassification matrix proposed by Huang (1998), is used to analyze the correspondence between clusters and the haplogroups or surname of the instances. Clustering accuracy is defined in Equation (13).

$$Clustering\ accuracy = \frac{\sum_{i=1}^{k} a_i}{n} \tag{13}$$

where k, is the number of clusters, a_i is the number of instances occurring in both cluster i and its corresponding haplogroup or surname and n is the number of instances in the data sets.

For precision and recall, the calculation is based on Equation (14) and (15) respectively.

$$Precision = \frac{\sum_{l=1}^{k} \left(\frac{a_l}{a_l + b_l}\right)}{n} \tag{14}$$

$$Recall = \frac{\sum_{l=1}^{k} \left(\frac{a_l}{a_l + c_l}\right)}{n} \tag{15}$$

where is the number of correctly classified objects; is the number of incorrectly classified objects; is the number of objects in a given class but not in a cluster; n is the number of classes/clusters.

Table 1.1 gives overview clustering results of the evaluated algorithms. The bold faced numbers refer to the best clustering result obtained by that particular algorithm. For Y-STR 535 dataset, the highest average clustering accuracy belongs to *k*-Modes algorithm. The algorithm obtained the average of clustering accuracy, 80.38% as compared to the other algorithms: *k*-Means (77.78%) and *k*-Medoids (78.19%). However, in contrast the *k*-Medoids algorithm produces a value that closes to zero for standard deviation. The algorithm also obtained the highest value of minimum accuracy of 100 runs, whereas the *k*-Modes algorithm recorded the highest value of 94.77% for maximum value of 100 runs.

For Y-STR 112 data set, the average clustering accuracy obtained by all algorithms is in between 38%-44% only. This is because all algorithms cannot work well with the objects having very strong similarity among the classes. In fact, some of the Y-STR objects are absolutely similar throughout 25 attributes (markers). However, the representative object-based technique produced the highest value of 43.63% but for the maximum value, the *k*-Means obtained about 47.32%. Overall results can be seen; the three clustering algorithms seem to be no significant difference as it merely differs about 2% -5% only.

Dataset	Evaluation (accuracy)	Hard Clustering Algorithms		
		k-Mean	*k*-Modes	*k*-Medoids
535 Y-STR	Average	0.7778	**0.8038**	0.7819
	Standard Deviation	0.0819	0.0922	**0.0262**
	Max	0.9402	**0.9477**	0.8336
	Min	0.6000	0.5925	**0.7514**
112 Y-STR	Average	0.3860	0.4212	**0.4363**
	Standard Deviation	0.0286	0.0265	**0.0149**
	Max	**0.4732**	0.4643	0.4554
	Min	0.3214	0.3393	**0.3482**

Table 1.1: THE SUMMARY RESULT FOR 100 RUNS OF FOUR ALGORITHMS

Table 1.2 and 1.3 give some insight values of precision and recall respectively for each algorithm. The precision and recall that are very close to 1 indicate the best matching for each pair of cluster and the corresponding class. Generally, most of the highest values for precision and recall are obtained by the *k*-Modes and the *k*-Medoids algorithms. The *k*-Modes algorithm initially dominates precision values, whereas the *k*-Medoids algorithm dictates the recall values.

Dataset	Evaluation (Precision)	Hard Clustering Algorithms		
		k-Means	*k*-Modes	*k*-Medoids
535 Y-STR	Average	0.6971	**0.7338**	0.6989
	Standard Deviation	0.0905	0.0890	**0.0575**
	Max	0.8838	**0.9000**	0.7839
	Min	0.4886	0.5387	**0.5444**
112 Y-STR	Average	0.3306	0.3857	**0.4196**
	Standard Deviation	0.0617	0.1064	**0.0351**
	Max	**0.4662**	0.6641	0.4889
	Min	0.1932	0.1934	**0.2010**

Table 1.2: THE SUMMARY RESULT FOR PRECISION

Dataset	Evaluation (Recall)	Hard Clustering Algorithms		
		k-Mean	k-Modes	k-Medoids
535 Y-STR	Average	0.7081	**0.7445**	0.6949
	Standard Deviation	0.0833	0.0905	**0.0480**
	Max	0.8745	**0.8827**	0.8569
	Min	0.5363	0.5202	**0.9988**
112 Y-STR	Average	0.3381	0.3332	**0.4826**
	Standard Deviation	0.0882	0.0792	**0.0484**
	Max	0.5075	0.4889	**0.6032**
	Min	0.1325	**0.2027**	0.1764

Table 1.3: THE SUMMARY RESULT FOR RECALL

From time efficiency point of view, it is obviously shown that the *k*-Medoids algorithm takes more time to handle partitioning Y-STR data set. The *k*-Medoids algorithm requires 10 – 13 minutes to partition Y-STR data set of 535 objects. Figure 2 shows the time taken in seconds for each algorithm, based on Y-STR 535 data set. Take note that the time is based on personal computers with AMD Athlon™ 64 X2 Dual Core Processor 6000+ with 3.00 GHz and 2.00 Gb memory. The lowest time recorded by the *k*-Means clustering algorithm, where the maximum time taken by the algorithm is only 11 seconds. However, the *k*-Modes algorithm recorded time between 15 – 37 seconds to complete a clustering process.

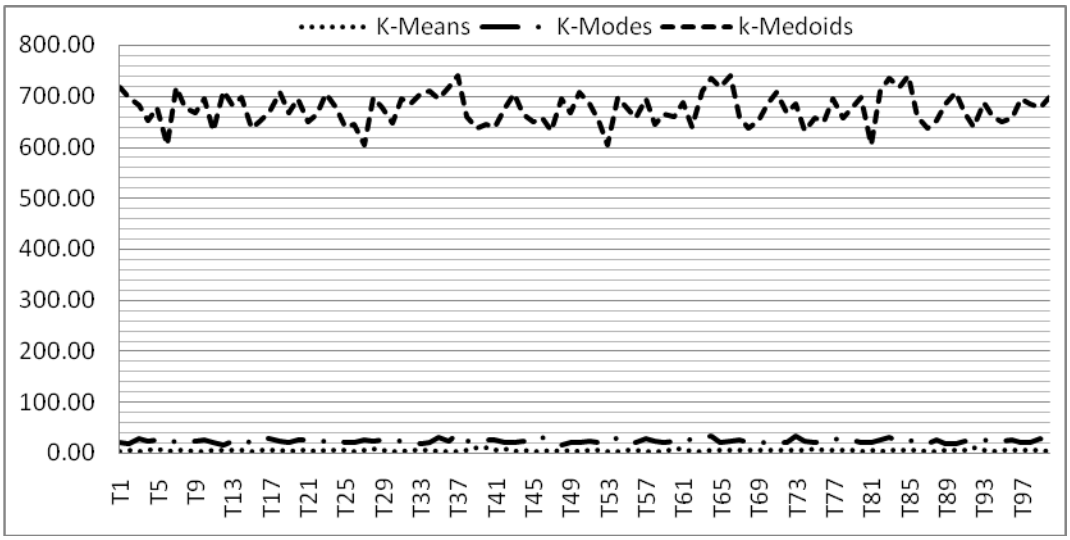

Figure 2**:** COMPARISON TIME TAKEN BY *k*-MEANS, *k*-MODES AND *k*-MEDOIDS

6. Conclusion

Overall results can be concluded that the centroid-based partitioning technique is the most reliable method in partitioning Y-STR data; even the results show that the average clustering accuracy is merely about 2%- 5% different among the three algorithms. In addition, the representative object-based partitioning technique causes high time consuming and its average clustering accuracy is also less than the *k*-Modes algorithm. If the overall results of the representative object-based partitioning technique showed that the average clustering accuracy was obviously better than the others, it could be tested for the other extended *k*-Medoids algorithms such as CLARA and CLARANS. These two algorithms are used for large data set and improved the time efficiency.

In the centroid-based partitioning technique, both algorithms seem to be an equal chance to be modified in order to improve the clustering accuracy of Y-STR data. However, from the results, it shows the *k*-Modes algorithm should be chosen first for further improvement. Furthermore, from the observation of Y-STR data, the patterns are made up of many occurrences, in which they can be treated as modes. In addition, the modal haplotypes that are used to measure the genetic distance is also based on the modes. However, the modal haplotypes are not necessarily modes for all cases in any given data set because the modal haplotypes are the established references by SNP methods for a group that shares a common ancestor.

In conclusion, the ideal case if the modal haplotypes can be used as the centroids, then the *k*-Modes algorithm could be improved in partitioning Y-STR data. However, given an arbitrary Y-STR dataset, there is no way to impose the modal haplotypes as its centroids because there is no specific formula to establish it from a given data set.

7. Acknowledgement

This research is part of our main research of the DNA kinship analyses funded by Fundamental Grant Research Scheme (FRGS), Ministry of Higher Education of Malaysia (Ref. no. 600-IRDC/ST/FRGS.5/3/1293; Project Code: 211201070005). Firstly, we thank Research Management Institute (RMI), Universiti Teknologi MARA (UiTM) Malaysia for their full support of this research. Secondly, we would like to extend our gratitude to many contributors toward the completion of this paper especially the dedication of our research assistance: Mr. Zahari, Miss Hasmarina, Mr. Syahrul and Miss Nurin.

References

1) 2007. Anthropological genetic: theory, methods and applications, edited by M.H. Crawford. Cambridge University Press.

2) Ali Seman, Zainab Abu Bakar & Azizian Mohd. Sapawi. 2010. Centre-based clustering for Y-Short Tandem Repeats (Y-STR) as Numerical and Categorical data. Proceedings 2010 Information Retrieval and knowledge management, Shah Alam, Malaysia. 28-33

3) Bezdek J. 1981. Pattern Recognition with Fuzzy Objective Function Algorithms. Plenum Press

4) D.W. Kim. K.Y. Lee. D. Lee. & K.H, Lee. 2005. *k*-populations algorithm for clustering categorical data. Pattern Recognition. 38, 1131–1134.

5) Donald Surname Project. Retrieved May 3, 2010, from http://dna-project.clan-donald-usa.org/

6) Family Tree DNA. Retrieved May 3, 2010, from www.familytreedna.com/public/IrelandHeritage/

7) Fitzpatrick C. & Yeiser, A. 2005. DNA & Genealogy. Rice Book Press, Fountain Valley, CA

8) Fitzpatrick C. 2005. Forensic genealogy. Rice Book Press, Fountain Valley, CA

9) Gan G. Ma C. & Wu, J. 2007. Data clustering: Theory, algorithms, and applications. Society for Industrial and Applied Mathematics (SIAM).

10) Gustafson D.E. & Kessel, W.C. 1979. Fuzzy clustering with a Fuzzy Covariance Matrix. Proceedings IEEE on Decision and Control. 761–766

11) Han J. & Kamber, M. 2001. Data Mining: concept and techniques.Morgan Kaufman Publisher, San Francisco.

12) International Society of Genetic Genealogy (ISOGG). Retrieved May 3, 2010, from http://www.isogg.org/

13) Kaufman L. & Rousseeuw, P.J. 1990. Finding groups in data: an introduction to cluster analysis. John Wiley & Sons: New York.

14) L. Kaufman & P.J, Rousseeuw. 1987. Clustering by means of medoids. Elsevier, 405-416.

15) M.K. Ng. M.J. Li. J.Z. Huang. & Z. He. 2007. On the impact of dissimilarity measure in k-modes clustering algorithm, IEEE Transactions of Pattern Analysis and Machine Intelligence. 29(3), 503-507.

16) M.K. Ng. & L. Jing. 2009. A new fuzzy *k*-modes clustering algorithm for categorical data. International Journal of Granular, Rough Sets and Intelligent Systems. 1(1), 105-119.

17) Macqueen J.B. 1967. Some methods for classification and analysis of multivariate observations. Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability, 281–297.

18) Ng R. & Han, J. 1994. Efficient and effective clustering methods for spatial data mining. Proceedings of the 20th international conference on very large databases, Santiago, Chile. 144-155.

19) Philips S. 2002. Acceleration of *k*-means and related clustering algorithms. In Mount, D and Stein, C, editors, ALENEX: International workshop on algorithm engineering and experimentation, LNCS. 2409, 166-177.

20) The Y-Chromosome Consortium (YCC). Retrieved May 3, 2010, from http://ycc.biosci.arizona.edu/

21) V. Faber. 1994. Clustering and the continuous *k*-means algorithm. Los Alamos Science. 22, 138-144

22) Worldfamilies.net. Retrieved May 3, 2010, from www.worldfamilies.net

23) Z. He. X. Xu. S. Deng. 2007. Attribute Value Weighting in k-Modes Clustering, Computer Science e-Prints: arXiv:cs/0701013v1 [cs.AI], Cornell University Library, Cornell University, Ithaca, NY, USA, http://arxiv.org/abs/cs/0701013v1, 1-15.

24) Z. Huang. 1998. Extensions to the *k*-Means algorithm for clustering large data sets with categorical values. Data Mining and Knowledge Discovery. 2, 283–304.