Knowledge-Based Systems 39 (2013) 133–143

Contents lists available at SciVerse ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Ranked *k*-medoids: A fast and accurate rank-based partitioning algorithm for clustering large datasets

Seyed Mohammad Razavi Zadegan [a], Mehdi Mirzaie [b,c], Farahnaz Sadoughi [a,*]

[a] Department of Health Information Management, School of Health Management and Information Sciences, Tehran University of Medical Sciences, Tehran, Iran
[b] Proteomics Research Center, Faculty of Paramedical Sciences, Shahid Beheshti University of Medical Sciences, Tehran, Iran
[c] Department of Bioinformatics, School of Computer Science, Institute for Research in Fundamental Sciences (IPM), Tehran, Iran

ARTICLE INFO

Article history:
Received 13 May 2012
Received in revised form 6 October 2012
Accepted 14 October 2012
Available online 23 November 2012

Keywords:
Clustering analysis
Partitioning clustering
k-Medoids clustering
k-Harmonic means
External validation measures

ABSTRACT

Clustering analysis is the process of dividing a set of objects into none-overlapping subsets. Each subset is a cluster, such that objects in the cluster are similar to one another and dissimilar to the objects in the other clusters. Most of the algorithms in partitioning approach of clustering suffer from trapping in local optimum and the sensitivity to initialization and outliers. In this paper, we introduce a novel partitioning algorithm that its initialization does not lead the algorithm to local optimum and can find all the Gaussian-shaped clusters if it has the right number of them. In this algorithm, the similarity between pairs of objects are computed once and updating the medoids in each iteration costs $O(k \times m)$ where k is the number of clusters and m is the number of objects needed to update medoids of the clusters. Comparison between our algorithm and two other partitioning algorithms is performed by using four well-known external validation measures over seven standard datasets. The results for the larger datasets show the superiority of the proposed algorithm over two other algorithms in terms of speed and accuracy.

1. Introduction

Clustering analysis is the process of dividing a set of objects into none-overlapping subsets. Each subset is a cluster, such that objects in the cluster are similar to one another (intra-similarity) and dissimilar to objects in other clusters (inter-dissimilarity) [1]. Clustering analysis has been widely used in many applications such as business intelligence [2,3], image processing [4,5], Web search [6], biology [7–9], and security [10].

There are several approaches of clustering such as hierarchical [11,12], partitioning [13], density-based [14], model-based [15,16] and grid-based [17]. Xu and Wunsch [18] provided a good survey on the clustering algorithms. In this paper our particular interest is in the partitioning approach that divides a dataset into $k < N$ clusters (N is size of dataset) such that every cluster has a center by which other members are determined according to their similarities. By using an iterative manner and re-computing the centers, algorithms of this approach attempt to find the best partitioning which has a high degree of intra-similarity and inter-dissimilarity. They mostly report the circle-shaped or Gaussian-shaped clusters because of assigning an object in the dataset to the most similar center.

One of the most well-known algorithms of partitioning approach is *k*-means that re-computes each of the new centers by averaging the dissimilarities (distances) of the assigned objects to the old centers [19]. Although, *k*-mean is simple and efficient algorithm, but it is sensitive to outliers and also its greedy nature makes it sensitive to initialization which may cause trapping in local optimum [20]. Other methods were designed based on *k*-means such as Fuzzy C-Mean (FCM), a fuzzy version of *k*-means [21], and *K*-Harmonic Means (KHMs) that uses a harmonic means for re-computing centers [22]. FCM is useful for datasets that the boundaries among clusters are not well-separated. However, it has problems with outliers and local optimum [18]. KHM solves the problems with initialization and outliers, but, it still suffers from local optimum [20].

To overcome local optimum problem, optimization algorithms like genetic algorithm [23], particle swarm optimization [24], ant colony [25], firefly algorithm [26] and artificial bee colony [27] were employed. They mostly use the cost function of *k*-means or *K*-Harmonic Means and by using their stochastic manner, attempt to escape from local optimum and find the best centers. However, they have brought their problems to the clustering analysis. As an example, PSO algorithm has two important phases which are exploration and exploitation. If these two phases perform incorrectly, for any reason, such as setting inappropriate parameters, PSO will not work out and cannot find the global optimum [28].

* Corresponding author. Tel.: +98 2188794300.
E-mail addresses: f-sadoughi@tums.ac.ir, sadoughi.f@gmail.com (F. Sadoughi).

0950-7051/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.knosys.2012.10.012

Another series of attempt to reduce sensitivity to outliers are developed by *k*-medoids algorithms. In these algorithms the center of cluster or medoid is always one of the objects in the dataset and through iterations the most centrally located objects are supposed to be found. One of the first algorithms of *k*-medoids clustering is Partitioned Around Medians (PAM) which deals with pairs of objects in the dataset [13]. The high computational time of PAM has impeded using this algorithm for large dataset [29]. Based on PAM another algorithm named CLARA was designed to deal with larger dataset [13]. This new algorithm selects multiple samples from the dataset and applies PAM over them. CLARA can handle larger datasets in contrast to PAM. However, some new problems have been appeared. If the samples do not involve the real centrally located objects, CLARA cannot find the pattern in dataset [30]. Therefore, determining the best sampling method and the size of the samples are new challenge for the algorithm. Moreover the computational time did not improve. CLARANS [31] is aimed at improving CLARA. It uses randomize policy in choosing pairs and makes use of the neighborhood to update the medoids. CLARANS and other algorithms based on PAM have considerable difficulty in computational time [30].

Newly designed algorithm by Park and Jun [29] is much efficient than PAM-based algorithms. In this paper, their algorithm is referred to as '*Simple and Fast k-medoids*'. A new medoid in this algorithm is selected by examining all the assigned members to the old medoid, and choosing one who increases intra-similarity more than others. Therefore, in each iteration the new set of medoids is selected with running time $O(N)$, where N is the number of objects in the dataset. *Simple and Fast k-medoids* is so sensitive to initialization which increases the possibility of trapping in local optimum. As Park and Jun [29] showed, the results vary considerably according to the methods of initialization.

In this paper we introduce a novel *k*-medoids algorithm which the initialization does not lead the algorithm to local optimum and can find all the Gaussian-shaped clusters. The remaining of the paper is organized as follows: Section 2 describes the partitioning approach and the local optimum definition. A rank-based similarity among objects in the datasets is introduced; it helps us to find the medoids faster. The hostility relation in human society is employed to explain the new concept. At the end of Section 2 our proposed algorithm is described and its behavior is analyzed in theory. Section 3 presents the experiments. Two algorithms of the partitioning approach of clustering which are *K*-Harmonic Means (KHM) and *Simple and Fast k-medoids*, are selected for comparison. Two challenging artificial datasets and five real datasets are used and in order to evaluate the results, four well-known external validation measures are explained in this section. The results are discussed in Section 4 and Section 5 makes conclusion.

2. Ranked *k*-medoids

2.1. Partitioning approach

Suppose a dataset $D = \{X_1, X_2, \ldots, X_N\}$ in which every object is a d-dimensional vector like $X_i = (x_{i1}, x_{i2}, \ldots, x_{id})$. In the partitioning approach of clustering, D is divided into $k < N$ clusters. Every cluster has a center which can be a hypothetical (centroid) or a real object (medoid) in D, and a similarity between objects and the centers determines the members of clusters. The similarity among objects can be defined using the Euclidean distance, correlation, cosine similarity, etc. In this paper the Euclidean distance in Eq. (1) is used to express the dissimilarity between objects like X_i and X_j.

$$\|X_i - X_j\| = \sqrt{\sum_{k=1}^{p} (x_{ik} - x_{jk})^2} \tag{1}$$

The definition of local optimum is relative to the approaches of clustering. For example, if we are analyzing a dataset, e.g. Fig. 1, that its clusters are spiral-shaped, ring-shaped or moon-shaped, we should not expect that an algorithm in the partitioning approach finds the clusters. Therefore, claiming that the algorithm was trapped in local optimum is not acceptable because the algorithms in the partitioning approach are not designed to find these kinds of clusters and they intrinsically report Gaussian-shaped clusters. Here the problem arising from selecting wrong approach, not trapping in local optimum. The meaning of the local optimum for partitioning approach is made clear by Definition 1.

Definition 1. Local optimum is a partitioning of a dataset that combined some real clusters and divide at least one cluster into several estimated clusters, in a situation that all the clusters are Gaussian-shaped and the right number of them is given.

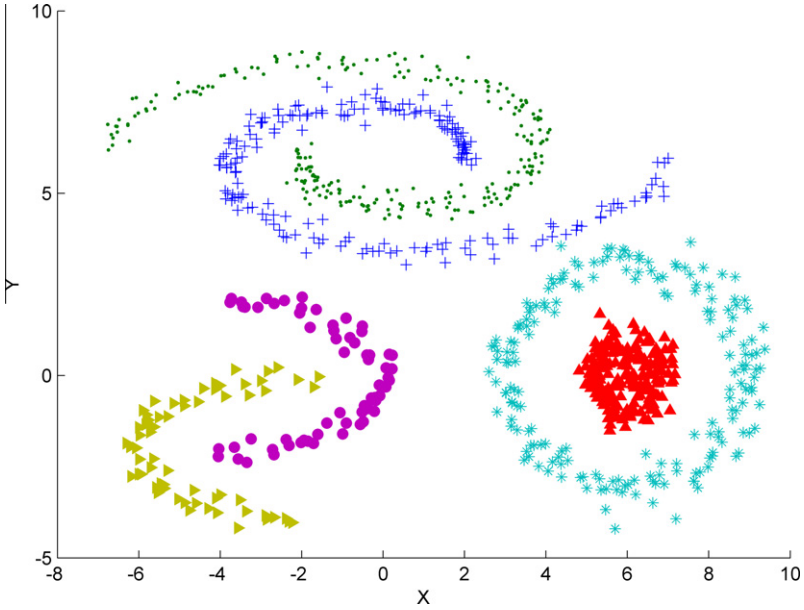

Fig. 1. A dataset of 2-dimensional points with six clusters. Algorithms of the partitioning approach cannot find the clusters.

Definition 1 is the appropriate definition for local optimum in the partitioning approach of clustering. It is essential for an algorithm of the partitioning approach to find the clusters of a dataset like what we mentioned in Definition 1. However, most of the well-known algorithms like KM, KHM and FCM have problem in finding the pattern of these suitable dataset [18,32].

2.2. Ranking and hostility relation

In our method the values of similarity between the objects are not directly used and we introduce new function that ranks objects according to their similarities. By this function, the more similar object gets lower rank. In other words, $rank(X_i, X_j) = l$ shows that X_j is the lth similar object to X_i among N objects in the dataset. The ranks of other objects according to an object like X_i can be computed by sorting the similarity values between X_i and other objects in the dataset. The rank function also expresses a rank matrix as follows:

$$R = [r_{ij}], \quad rank(X_i, X_j) = r_{ij}, \quad \forall\, X_i,\ X_j \in D \tag{2}$$

Since two objects are not always at the same rank of each other, R is not necessarily a symmetric matrix, in other words, it may happen that $rank(X_i, X_j) \neq rank(X_j, X_i)$.

R is similar to the hostility relation in human society. As we know, in human society each person ranks others according to their common interests or ideas and based on this system of ranking finds some persons friendly and others unfriendly. But, these feelings are not always symmetry, in other words, a person may be deeply hostile to somebody, but, he/she may not be that much hostile toward the first person. In our algorithm two objects are figuratively hostile toward each other according to their ranks, and the more similar objects have less hostile feelings toward each other. Thus, R is a matrix that shows the hostility relationship among objects in the dataset, and also, it contains the strength of these relations through numbers, from 1 (for the least hostility) to N (for the extreme hostility).

Similarly, from $r_{ij} < r_{ji}$ which is a case of asymmetric relation in R, can be understood that the rank of x_i is higher, according to the similarities of x_j to other objects (r_{ji}), than the rank of x_j according to the similarities of x_i to other objects (r_{ij}). Thus, it can be considered that x_j are much hostile or unfriendly toward x_i. Consequently, it reveals the existence of some other objects more similar to x_j that make x_i looks more unfriendly. We utilize this information in updating the medoids.

In order to find the medoids we introduce another quantity which is the hostility value (hv) of an object in a group of objects.

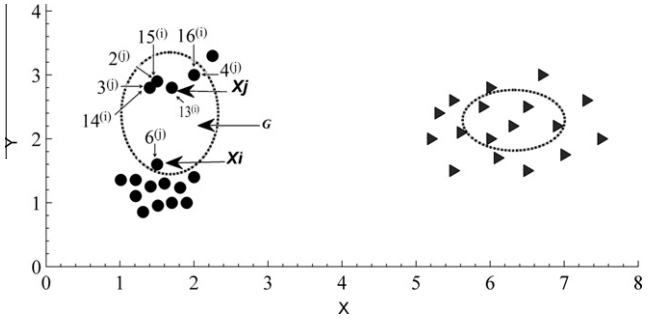

Fig. 2. Two clusters of objects in 2-dimensional space, and the Euclidean distance determined their dissimilarity. Groups of objects are enclosed by ellipses. Left cluster: The lowest object in the group (X_i) are more similar to the objects from the outside of the group, therefore, the ranks of the objects in the group increase and consequently the hostility value of X_i becomes maximum. Right cluster: All the objects in the group are approximately at the same rank of each other; therefore, their hostility values are nearly equal.

The hostility value of an object like X_i in a set of objects like G is defined as follows:

$$hv_i = \sum_{X_j \in G} r_{ij} \tag{3}$$

As an example, we compute the hostility value of two objects (X_i, X_j) in Fig. 2 in which the members of the group are enclosed by an ellipse and the Euclidean distance determines the dissimilarity among them. According to the similarity of X_i to the other objects (Right cluster is ignored), we can visually find out that the inner objects in the group are 13th to 16th similar objects to X_i (the numbers are distinguishable by label (i)). Thus, the hostility value of X_i is computed as follows:

$$hv_i = 1 + 13 + 14 + 15 + 16 = 59$$

Similarly, from the similarity between X_j and other objects can be understood that the inner objects in the group are 2nd, 3rd, 4th and 6th similar objects to X_j (the numbers are distinguishable by label (j)), therefore, the hostility value of X_j is computed as follows:

$$hv_j = 1 + 2 + 3 + 4 + 6 = 16$$

hv shows figuratively, how much an object is unfriendly with others in a group. For example, X_i in Fig. 2 is much hostile to the others rather than any other members. Moreover, the hostility values of a group provide clues about how objects in the group or even out of the group are scattered in the input space. The distribution of hvs in the group involves two possible cases. In the first case, the hostility values of data objects in a group are not nearly equal. Therefore, we can conclude that some objects are more hostile than others to the members of group. We see this heterogeneity in hvs because the objects which have greater hv are more similar to some objects from the outside of the group. Therefore, the outer objects are placed in lower rank and the inner ones place in higher rank, as left cluster in Fig. 2 shows. In the second case, the hostility values of objects in a group are approximately equal. It shows that the most of the similar objects are part of the group; thus, there are not many other similar objects from the outside of the group, as right cluster in Fig. 2 shows. Therefore, by using hvs of a group we can slightly discover how objects in the group or those around the group are scattered.

2.3. Proposed algorithm

Our proposed algorithm consists of the following steps:

Step 1. (Initialization)
- 1.1. Calculate the similarities among pairs of objects based on the similarity metric (e.g. Euclidean, correlation).
- 1.2. Calculate R matrix by sorting the similarity values and store the indexes of similar objects from the most similar to the least similar in *sorted index* matrix.
- 1.3. Select k medoids randomly.

Step 2. (Update medoids)
- 2.1. Do the following steps for each medoid:
  - 2.1.1. Select the group of the most similar objects to each medoid, using *sorted index* matrix. (The number of members of the group is determined by an input parameter m).
  - 2.1.2. Calculate the hostility values of every object in those groups using Eq. (3).
  - 2.1.3. Choose object with the maximum hostility value as the new medoid.

2.2. Relocate one of the medoids placed in the same group.
2.3. Go to step 2.1. until the maximum iteration will be reached.

Step 3. (Labeling objects)
3.1. Assign each object to the most similar medoid.

In each iteration, calculating k new medoids costs $O(k \times m)$ where shows the number of members in the group. The parameter m has a direct effect on the speed of the algorithm. By having a smaller group, the medoids approach the centers with the smaller steps. The medoids may also jump between clusters easily, if the group is too large. But, selecting m between 5 and 15 will be efficient for small and large datasets.

In the step 1.2. a matrix called *sortedIndex* matrix is returned from calculating the ranks of objects. It keeps the indexes of similar objects in the dataset. $sortedIndex_{ij} = t$, shows that the jth similar objects to X_i is X_t. By using *sortedIndex* matrix, the computational time of forming group G will be $O(m)$. To understand thoroughly this matrix, suppose a dataset D with one-dimensional points as follows:

$$D = \{X_1, X_2, X_3, X_4\} = \{1, 3, 4, 6\}$$

The matrix R and *sortedIndex* for D are as follows (closeness shows the similarity):

$$R = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 3 & 1 & 2 & 4 \\ 4 & 2 & 1 & 3 \\ 4 & 3 & 2 & 1 \end{bmatrix}; \quad sortedIndex = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 1 & 4 \\ 3 & 2 & 4 & 1 \\ 4 & 3 & 2 & 1 \end{bmatrix}$$

In the matrix R, $R_{21} = 3$ means that X_1 is the third similar object to X_2 and $sortedIndex_{34} = 1$ means that X_1 is the fourth similar object to X_3.

In order to understand how the algorithm escapes from local optimum, we will show how it moves the medoids through iterations and afterwards, analyzing the behavior of the algorithm in dealing with a dataset will be described.

2.3.1. Seeking the center

Suppose a cluster like C that its objects are scattered in input space by a Gaussian distribution with the average μ and the standard deviation σ. For selecting a new medoid, a group of the closest objects to the old medoid like G is needed. Imagine that G contains two object X_i and X_j such that X_i is closer to μ or the center of cluster, therefore we have:

$$X_i, X_j \in G, \quad \|X_i - \mu\| < \|X_j - \mu\| \tag{4}$$

We can also rewrite Eq. (3) as follows:

$$hv_i = \left(\frac{m(m+1)}{2}\right) + \sum_{\forall X_j \in G} |S_{ij}| \tag{5}$$

m is the number of objects in G and $|S_{ij}|$ is the number of objects in S_{ij} which is defined as follows:

$$S_{ij} = \{X_k \notin G | \|X_k - X_i\| < \|X_i - X_j\|\} \tag{6}$$

By using Eq. (3), the calculated ranks of every objects in the group (according to their similarities to X_i) are added to obtain hv_i. However in Eq. (5), at first the objects in the group are supposed to be the first m ranks of X_i, and afterwards the number of more similar objects to X_i, from the outside of the group are added. Now, an example is served to illustrate this new form of calculating hv. There are six points in Fig. 3 in which three of them are in the group G ($m = 3$). To calculating the hv_1 according to Eq. (3), the ranks of X_2 and X_3 are needed. The ranks of the objects in Fig. 3 considering their distances to X_1 are as follows:

Objects	X_1	X_2	X_3	X_4	X_5	X_6
Rank	1	4	6	5	3	2

Therefore, the hostility value of X_1 according to Eq. (3) is as follows:

$$hv_1 = 1 + 4 + 6 = 11$$

In order to calculate the hv_1 according to Eq. (5), the sets S_{12} and S_{13} should be identified. S_{12} is the set of objects from outside of the group which are closer than X_2 to X_1. Therefore, the set S_{12} is as follows:

$$S_{12} = \{X_5, X_6\}$$

Similarly, the members of S_{13} are:

$$S_{13} = \{X_4, X_5, X_6\}$$

According to Eq. (5) the hv_1 is calculated as follows:

$$hv_1 = \left(\frac{3 * 4}{2}\right) + (|S_{12}| + |S_{13}|) = (1 + 2 + 3) + (2 + 3) = 11$$

The distribution of C is Gaussian; therefore, the probability of objects closer to average is greater than those farther from average. It means that the probability of objects in S_{ij} (which is a subset of objects in the cluster) increases as X_i becomes closer to the average. Therefore, it reveals that S_{ij} is larger for X_i which is closer to the average than S_{ij}, for X_j which is farther than the average. Consequently, *hv*s in G increase as the objects approach the average.

In each iteration, our algorithm selects the maximum *hv* in G as the next medoid. We showed that the maximum *hv* in G occurred among those objects which are closer to the average or the center of cluster. Therefore, through iteration, the medoid distances from the edge of cluster and comes near the center. When the medoid is located in the center of cluster it never returns to the edge of cluster, because, the *hv*s in G decreases as the object comes near to the edge of the clusters. Therefore, the objects from the edge cannot have maximum *hv* and be the next possible medoid, in the presence of some objects near the center.

Fig. 4 shows the behavior of the algorithm in selecting the medoid, through four iterations. The only cluster in the figure is Gaussian-shaped and the current and the previous medoids are distinguishable by filled square and circle. The number of members in G is equal to 15, and the circles around these points make them recognizable. It can be seen how medoid approaches the center from the edge of the cluster. Finally, by locating at the center in

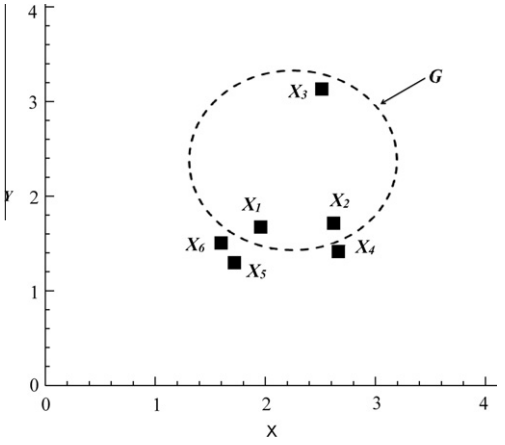

Fig. 3. Six two-dimensional points. X_1, X_2 and X_3 are in the group indicated by G.

the fourth iteration, the medoid and the group G will not leave there.

The behavior of the algorithm in leading medoid to the center of cluster can be illustrated from theoretical point of view. Therefore, suppose that the objects of cluster C follow one-dimensional (for the sake of simplicity) Gaussian distribution with parameters μ and σ. The density function of this distribution is as follows:

$$F(X) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(X-\mu)^2}{2\sigma^2}} \tag{7}$$

Fig. 5 shows the curve of a Gaussian distribution with the average μ. It can be seen that the probability of an objects closer to μ is greater than the farther one ($F(X_i) > F(X_j)$). The hachured area in Fig. 5A is the probability of objects which are in the cluster and closer than X_j to X_i. U_{ij} is a set of these objects:

$$U_{ij} = \{X_t \in C | \|X_t - X_i\| \leqslant \|X_j - X_i\|\} \tag{8}$$

This probability is directly related to the probability of an object belonged to the set S_{ij}, because S_{ij} is a subset of U_{ij}. Therefore, as X_i moves towards μ, the probability of objects like $X_k \in S_{ij}$ increases. Consequently, for two objects like X_i and X_j in the group G, the hv_i is greater than hv_j, if X_i is closer to μ than X_j.

2.3.2. Escaping from local optimum

After describing the movements of medoids, it is time to look at the behavior of the algorithm in dealing with a dataset containing k Gaussian-shaped clusters. The algorithm chooses k objects as the medoids of the clusters. Regarding to Definition 1, we will show that the algorithm can escape from local optimum.

Suppose a state at the early iterations of the running algorithm that most of the medoids share some clusters. As we have the right number of clusters, there are also some of them without medoid. We call the clusters with at least one medoid "discovered cluster" and those without medoid "undiscovered cluster" (Fig. 6). We showed that in a finite number of iterations medoids will be placed at the center of discovered clusters. In that moment, the medoids of a discovered cluster are so close that they are placed in their similar groups (a group which is needed to compute the hvs and find the new medoid). Then, the algorithm relocates all the medoids except one of them (step 2.2). Fig. 7 shows a situation that two medoids are the parts of their similar groups; therefore, one of them should be relocated. Afterwards, the relocated medoid might be placed inside an undiscovered cluster or even in another discovered cluster. In the first case, the medoid reaches the center in a finite number of iteration and possesses the undiscovered cluster. In the second case that there is a cluster with more than one medoid, a new relocation will happen. Obviously, the algorithm can be implemented such that these relocations of misplaced objects end, in a finite number of iterations (at the worst case it tests all the objects except the members of the undiscovered cluster). Therefore in a finite number of iterations, the algorithm can find all the clusters regardless of how the medoids were located initially.

Video 1 shows how our algorithm finds the clusters. In this video the medoids (big black circles) are plotted in each iteration, with the rest of the objects in the dataset. At early iterations (1–30) most of the medoids are placed in the same clusters and their separations are visible. As iteration continues, each medoid possesses a cluster of objects. In 41th iteration all the medoids are placed at the centers of clusters, therefore, nothing will happen except the little movements of medoids in the centers.

To expedite the process of finding real medoids, we can also double the size of similar groups and select half of them randomly to find next medoids. By this policy we enlarge the groups and consequently the steps to the centers of clusters as we keep the number of members unchanged.

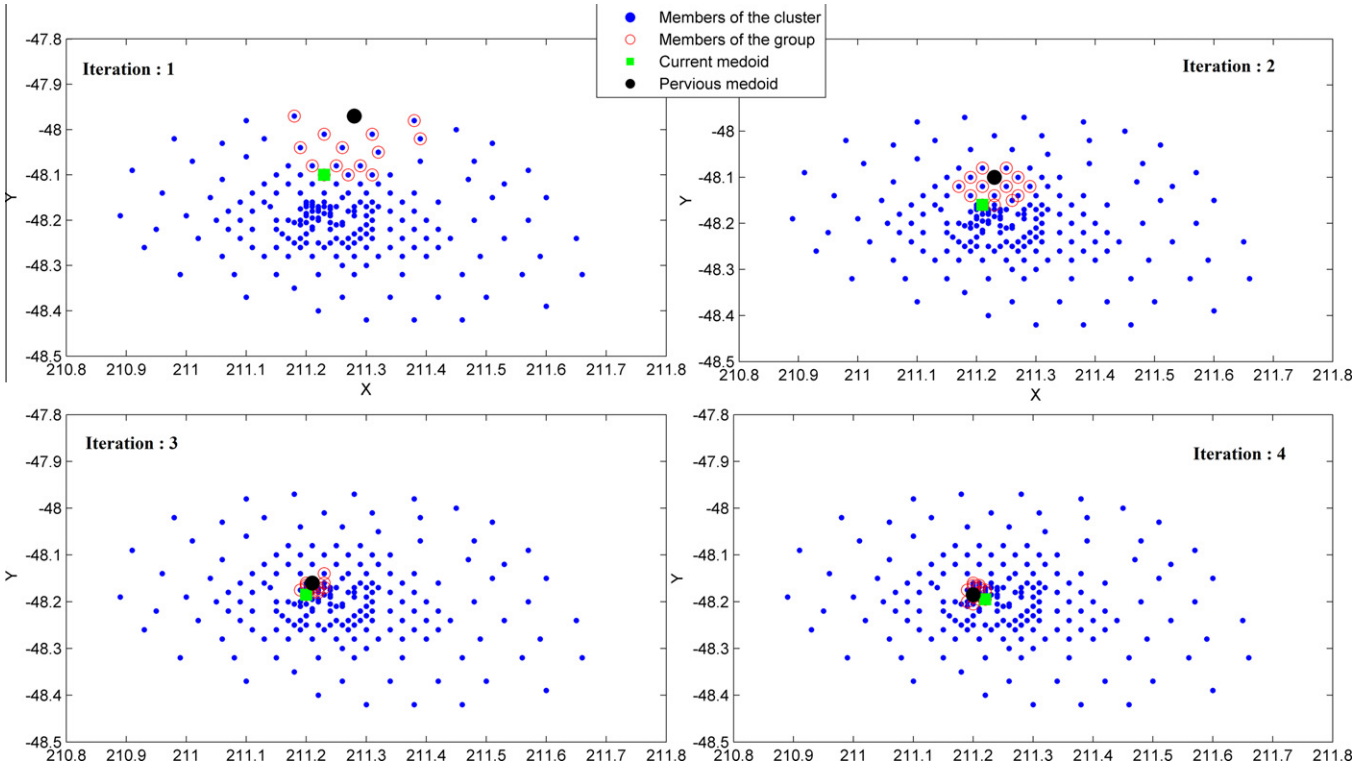

Fig. 4. Four iterations of RKM algorithm. The current and previous medoid with the members of group G have been indicated at the end of each iteration. Initial medoid is placed at the edge of cluster, but, in the fourth iteration it has found the center.

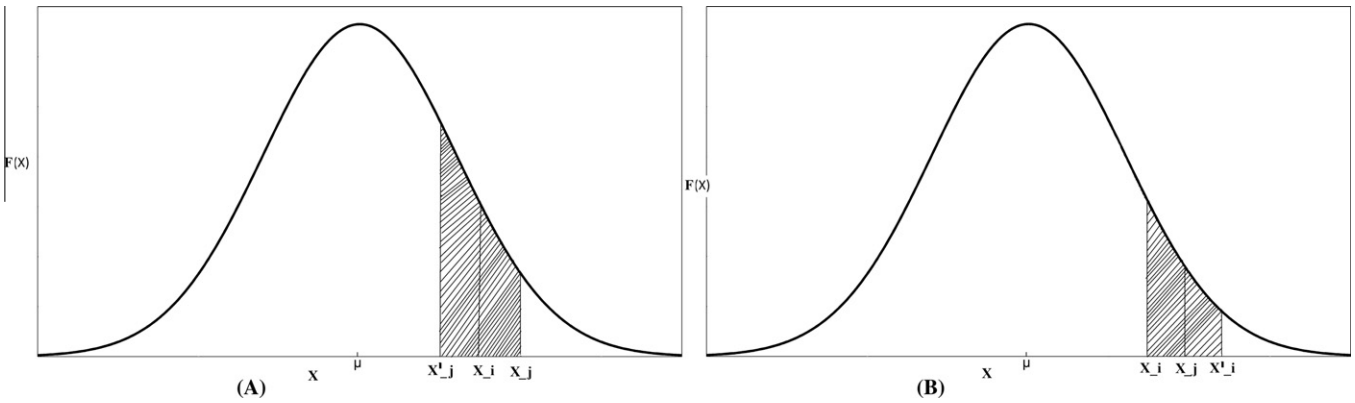

Fig. 5. The curve of Gaussian distribution with the average (μ). (A) Hachured area shows the probability of a point belonging to U_{ij} and $\|X'_j - X_i\| = \|X_j - X_i\|$ and (B) hachured area shows the probability of a point belonging to U_{ji} and $\|X'_i - X_j\| = \|X_j - X_i\|$.

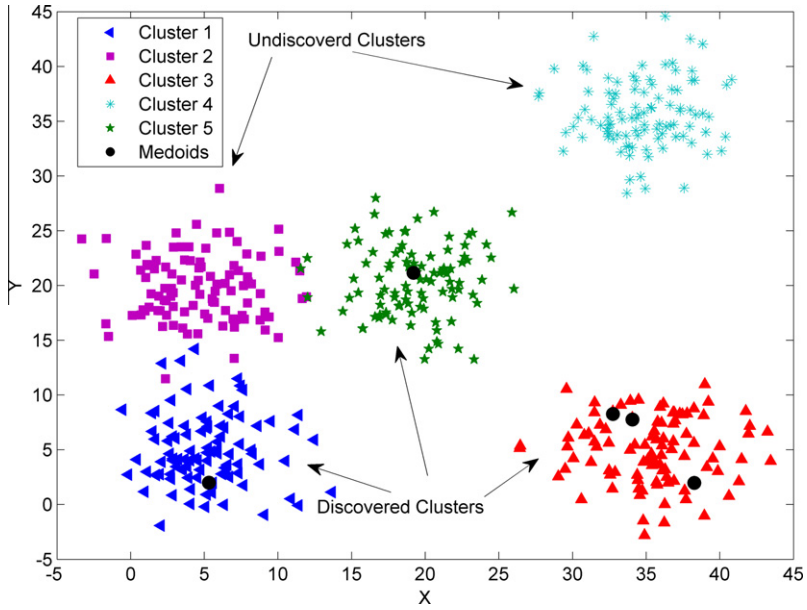

Fig. 6. Five clusters and the initial location of five medoids are shown. Three clusters have at least one medoid (discovered clusters), and two of them are without any medoid (undiscovered clusters).

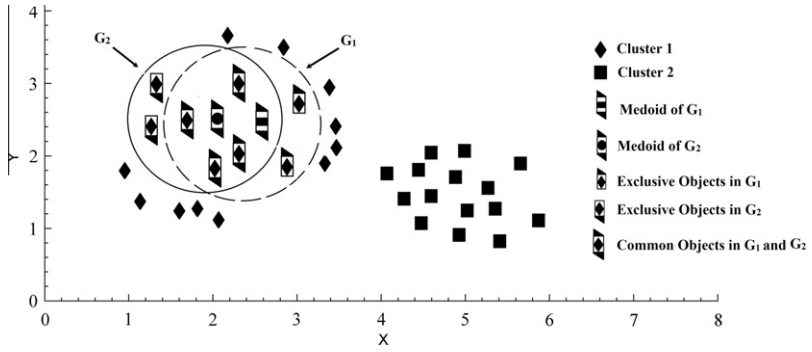

Fig. 7. Two clusters of objects. The diamond-shaped cluster has two medoids which are the members of their similar groups and one of them should leave the cluster.

3. Experiments

3.1. Algorithms

Our proposed algorithm was compared with two other partitioning algorithms, *K*-Harmonic Means (KHMs) and *Simple and Fast k-medoids*. KHM is one of the most well-known algorithms in this family that unlike *k*-means is not sensitive to initialization and outliers. Updating k centers, in each iteration, costs $O(k \times N \times D)$ in a situation that the Euclidean distance shows dissimilarity among objects of the dataset which are D-dimensional.

Simple and Fast k-medoids like our algorithm can compute the similarity among objects in the dataset once and then can find the best possible partitions. It performs better than *k*-means [29] so it can be a good comparison between this algorithm and our algorithm in term of efficiency. This algorithm updates the new medoids in $O(N)$ where N shows the number of objects in the dataset.

The parameters which are maximum iterations (*maxiter*) and the number of members in the group (m) are presented in Table 1. All the algorithms are run on a computer with windows operating system, Intel Core2Duo 2.66 GHz CPU and 4 GB RAM. Each algorithm was run 20 times over seven datasets. These datasets will be described at the next section. To evaluate these algorithms, four validation measures were used. The definition of these measures and the meaning of their values are presented at Section 3.3. Finally, the results are brought in Table 2 in the form of *average* (*standard deviation*). Time in Table 2 shows the average time of running algorithms in second.

3.2. Datasets

We have employed two artificial datasets and five real datasets to perform experiments. In the following these datasets are described briefly.

3.2.1. A_2

This dataset contains 5250 2-dimensional points separated in 35 clusters. As Fig. 8 shows, A_2 with its circle shape clusters is ideal for partitioning approach clustering. But the large number of clusters increases the possibility of trapping in local optimum. Therefore, it can be a good challenge for these kinds of algorithms to find the real pattern. A_2 is brought in Supplementary and can be found in [33].

3.2.2. S_3

This dataset contains 5000 2-dimensional points separated in 15 clusters (Fig. 9). Clusters in S_3 have more overlaps than A_2 and the boundaries of clusters are not well-separated which makes it more challenging. Moreover, there are some dispersed clusters among dense ones and they form a difficult pattern. Here again, being sensitive to initialization and trapping in local optimum can create troublesome pattern recognition. S_3 was used by Franti and Virmajoki [34] and it is brought in our Supplementary and [33].

The following datasets are well-known and widely used in many studies. These datasets are stored in UCI repository [35].

3.2.3. Multiple features (mfeat-fac)

This dataset contains 2000 objects which are handwritten numerals ('0'⋯'9') extracted from a collection of Dutch utility maps. The objects are equally separated in 10 classes. These objects are described by different feature sets. In this experiment we use profile correlations feature set that consists of 216 features.

3.2.4. Wisconsin Breast Cancer (WBC)

This dataset contains 699 objects described by nine numerical features which are clump thickness, cell size uniformity, cell shape uniformity, marginal adhesion, single epithelial cell size, bare nuclei, bland chromatin, normal nucleoli, and mitoses. We removed 16 objects of this dataset because of having missing values, so, its size decreased to 683 objects grouped in two classes of benign and malignant.

Table 1
The parameters of running ranked *k*-medoids over different datasets.

Parameters	Datasets						
	A2	S3	mfeat-fac	WBC	Iris	Zoo	Wine
m	10	10	10	10	5	5	5
maxiter	100	100	50	50	50	50	50

Table 2
The results of running the algorithms 20 times for each datasets are presented by *average* (*standard deviation*) format. The best results are in the bold format.

	Simple and Fast k-medoids	KHM	Ranked *k*-medoids
A2			
ARI	0.777 (0.044)	0.831 (0.04)	**0.950 (0.025)**
F-measure	0.841 (0.034)	0.892 (0.027)	**0.971 (0.015)**
Purity	0.834 (0.035)	0.886 (0.029)	**0.970 (0.017)**
Mirkin	0.014 (0.003)	0.010 (0.003)	**0.003 (0.001)**
Time	2.715 (0.879)	4.369 (0.017)	**0.950(1.398)**
S3			
ARI	0.635 (0.057)	0.680 (0.029)	**0.721 (0.019)**
F-measure	0.778 (0.047)	0.819 (0.023)	**0.852 (0.015)**
Purity	0.764 (0.051)	0.813 (0.026)	**0.850 (0.018)**
Mirkin	0.049 (0.011)	0.041 (0.004)	**0.035 (0.003)**
Time	3.764 (1.643)	1.786 (0.012)	**0.528 (1.253)**
mfeat-fac			
ARI	0.429 (0.054)	0.287 (0.003)	**0.439 (0.045)**
F-measure	0.628 (0.049)	0.472 (0.013)	**0.637 (0.048)**
Purity	0.619 (0.048)	0.374 (0.009)	**0.621 (0.056)**
Mirkin	0.110 (0.012)	0.192 (0.003)	**0.108 (0.012)**
Time	0.966 (3.268)	17.563 (0.022)	**0.932 (3.467)**
WBC			
ARI	0.657 (0.271)	0.710 (0.245)	**0.730 (0.191)**
F-measure	0.902 (0.088)	0.930 (0.048)	**0.931 (0.041)**
Purity	0.896 (0.101)	0.916 (0.089)	**0.924 (0.068)**
Mirkin	0.168 (0.129)	0.139 (0.108)	**0.131 (0.086)**
Time	3.133 (1.216)	0.218 (0.319)	**0.044 (0.092)**
Iris			
ARI	0.651 (0.149)	**0.729 (0)**	0.665 (0.157)
F-measure	0.832 (0.109)	**0.894 (0)**	0.854 (0.090)
Purity	0.830 (0.110)	**0.893 (0)**	0.848 (0.099)
Mirkin	0.163 (0.079)	**0.119 (0)**	0.154 (0.082)
time	**0.002 (0.001)**	0.040 (0)	0.042 (0.016)
Zoo			
ARI	0.519 (0.119)	0.509 (0.089)	**0.547 (0.145)**
F-measure	0.727 (0.067)	0.708 (0.057)	**0.727 (0.085)**
Purity	**0.797 (0.035)**	0.816 (0.026)	0.787 (0.068)
Mirkin	0.165 (0.044)	0.155 (0.026)	**0.152 (0.049)**
time	**0.001 (0.001)**	0.079 (0.001)	0.062 (0.019)
Wine			
ARI	**0.374 (0.016)**	0.354 (0)	**0.374 (0.035)**
F-measure	0.695 (0.013)	0.672 (0)	**0.698 (0.020)**
Purity	**0.704 (0.014)**	0.685 (0)	0.703 (0.020)
Mirkin	0.287 (0.019)	0.288 (0)	**0.285 (0.028)**
Time	**0.003 (0.001)**	0.055 (0.001)	0.042 (0.016)

3.2.5. Iris

This dataset contains 150 objects described with four features and are equally separated in three clusters. Objects in this dataset are Iris plants and the four numerical features are sepal length, sepal width, petal length and petal width.

3.2.6. Zoo

It consists of 101 objects described by one numerical feature which is leg and 15 boolean features which are hair, feathers, eggs, milk, airborne, aquatic, predator, toothed, backbone, breathes, venomous, fins, legs, tail, domestic and catsize. Zoo is categorized in seven classes.

3.2.7. Wine

This dataset consists of 178 objects described by 13 features. The instances of the dataset are the result of chemical analyses

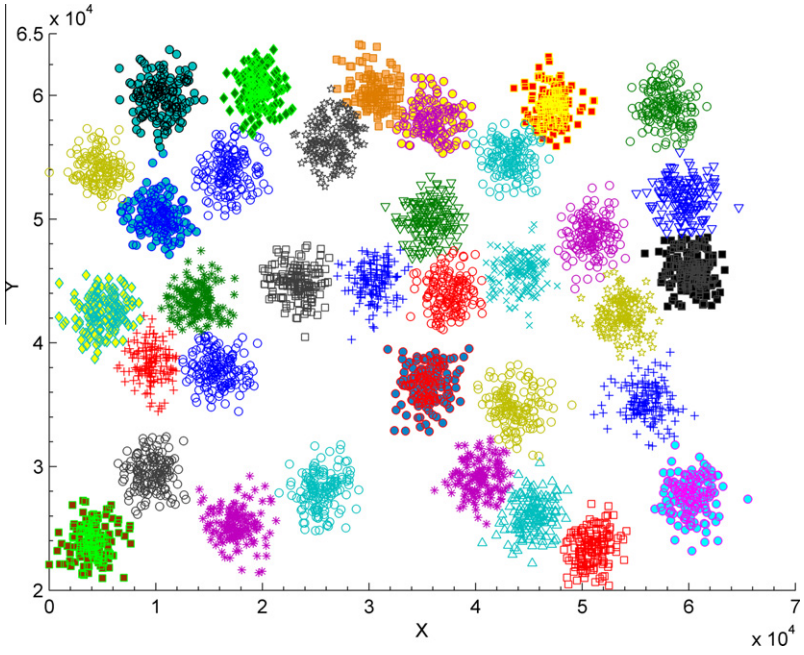

Fig. 8. A2.

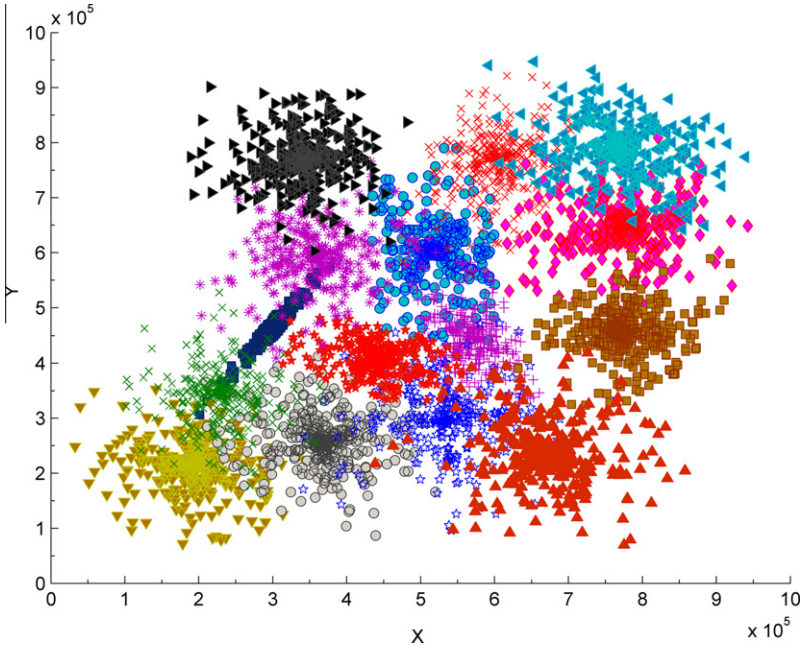

Fig. 9. S3.

of wine in Italy. The numerical feature are alcohol, malic acid, ash, alcalinity of ash, magnesium, total phenols, flavanoids, nonflavanoid phenols, proanthocyanins, color intensity, hue, OD280/OD315 of diluted wines and proline. Wine is classified in three classes.

3.3. Evaluation functions

To evaluate the results of clustering methods, two types of cluster validation techniques are using: internal validation measures and external validation measures. Internal validation measures evaluate clusters based on the structure of the dataset; therefore, intra-clusters similarity (cohesion) and inter-cluster dissimilarity (separation) are the main factors for these measures. On the other hand, external validation measures evaluate clusters by comparing computed labels of objects with their real label [36]. In our experiments, the clusters were evaluated by four external validation measures which are Mirkin [37], Purity [38], *F*-measure [36] and Adjusted RandIndex (ARI) [39]. In the following these measures are described briefly.

To explain the external validation measures we need a matrix called association matrix. Suppose that a dataset with N objects is partitioned into $C = \{c_1, c_2, \ldots, c_I\}$ classes and the algorithm finds $K = \{k_1, k_2, \ldots, k_J\}$ clusters, then matrix $A = [a_{ij}]_{I \times J}$ is association

matrix where a_{ij} indicates number of c_i's members which belong to k_i. Moreover, a_i and a_j as marginal numbers is defined by:

$$a_{.j} = \sum_i a_{ij} \tag{9}$$

$$a_{.i} = \sum_j a_{ij} \tag{10}$$

3.3.1. Mirkin

The Mirkin metric is defined by Eq. (11) and related to the number of disagreed pairs in the sets K and C. Mirkin metric is scaled in the interval [0,1] which is 0 for identical sets of labels or complete finding of the pattern and positive otherwise.

$$\text{Mirkin}(C,K) = \frac{1}{N^2}\left(\sum_{i=1}^{I} a_{i.}^2 + \sum_{j=1}^{J} a_{.j}^2 - 2\times\sum_{j=1}^{J}\sum_{i=1}^{I} a_{ij}^2\right) \tag{11}$$

3.3.2. Purity

The purity of a cluster is defined as follows:

$$\text{Purity}(k_p) = \frac{1}{a_{.p}}(\max(a_{ip})), \quad i = 1,\ldots,I \tag{12}$$

The above purity determines the class that was predicted by k_p. The larger value of Eq. (12) shows more common objects in cluster k_p and the predicted class. The purity of whole solution can be defined by using a weighted mean of the individual cluster purities.

$$\text{purity}(K) = \sum_{p=1}^{J}\left(\frac{a_{.p}}{N}\times\text{Purity}(k_p)\right) \tag{13}$$

3.3.3. F-measure

It is widely used in information retrieval and is based on two concepts of *precision* and *recall* which indicate accuracy and coverage of retrieved information correspondingly, and are defined by:

$$\text{precision}(k_i,c_j) = \frac{a_{ij}}{a_{.j}} \tag{14}$$

$$\text{recall}(k_i,c_j) = \frac{a_{ij}}{a_{i.}} \tag{15}$$

F-measure of cluster k_i regards to class c_j is the harmonic mean of its precision and recall. It has a low value for a situation in which recall and precision are unbalanced. This value is defined by:

$$F_{kc}(k_i,c_j) = \frac{2}{1/\text{precision}(k_i,c_j) + 1/\text{recall}(k_i,c_j)} \tag{16}$$

Consequently, F-measure of a cluster is defined by:

$$F_k(k_p) = \max_{j=1,2,\ldots,I} F(k_p,c_j) \tag{17}$$

Therefore the F-measure of whole clustering solution can be defined as Eq. (18) and the greater its value, the better clustering solution.

$$F(K) = \sum_{p=1}^{J}\frac{a_{.p}}{N}F(k_p) \tag{18}$$

3.3.4. Adjusted Rand Index (ARI)

It is one of the most popular external validation measures ranging from 0 to 1. Higher value of ARI shows better quality of clusters. To explain ARI four quantities based on pairs of objects are defined: 'a' is the number of pairs which are in the same class in C and in the same cluster in K, 'b' is the number of pairs which are in the same class in C but not in the same cluster in K, 'c' is the number of pairs which are in the different classes of C but in the same cluster in K and 'd' is the number of pairs placed on different classes and clusters in C and K. According to these quantities rand index (RI) is defined by:

$$\text{RI}(C,K) = \frac{a+d}{a+b+c+d} = \frac{a+d}{\binom{N}{2}} \tag{19}$$

RI is not uniformly distributed in the interval [0,1] and cannot show properly the differences between random partitioning and real partitioning [40]. ARI is an adjustment of RI that solves this problem and is defined by:

$$\text{ARI}(C,K) = \frac{RI(C,K) - E(RI)}{1 - E(RI)} \tag{20}$$

$E(RI)$ is the expected value of rand index under the generalized hypergeometric assumption. ARI can be simplified and rewritten as follows:

$$\text{ARI}(C,K) = \frac{\sum_{i=1}^{I}\sum_{j=1}^{J}\binom{a_{ij}}{2} - \left[\sum_{i=1}^{I}\binom{a_{i.}}{2}\sum_{j=1}^{J}\binom{a_{.j}}{2}\right]\Big/\binom{N}{2}}{1/2\left[\sum_{i=1}^{I}\binom{a_{i.}}{2} + \sum_{j=1}^{J}\binom{a_{.j}}{2}\right] - \left[\sum_{i=1}^{I}\binom{a_{i.}}{2}\sum_{j=1}^{J}\binom{a_{.j}}{2}\right]\Big/\binom{N}{2}} \tag{21}$$

Some simple examples and comparison between RI and ARI are brought in [40].

4. Discussion

There is a striking contrast between the results of our method and the other algorithms for datasets A_2, S_3, mfeat-fac and WBC. Worse values of external validation measures for KHM and *Simple and Fast k-medoids* reveal that they have combined some classes or divide one class into several clusters and trapped in local optimum. For the smaller datasets including Iris, zoo and wine, our algorithm does not outperform the two other algorithms. However, the close values of external validation measures for these datasets show that RKM has not lost the pattern, and some misplaced objects in the edge of the clusters make RKM slightly worse than other algorithm.

To understand why these results emerged from experiments, the behavior of the algorithms should be considered. Algorithms like KHM or *Simple and Fast k-medoids* have tendency to be stationary. By the stationary, we mean that the algorithm updates the centroid of a cluster, if a more centrally located point can be found, such that, it decreases the cost function of the algorithm. This greedy behavior is one of the main reasons of trapping in the local optimum. Therefore, as the number of clusters and objects increase, the local optimum situations and the probability of trapping in the local optimum raise too (as we have seen in the result of dataset A_2 and S_3). But, in the smaller datasets with the decreased number of local optimum, they can be placed at the optimum point by the help of a suitable initialization, and can keep the point till the end. Therefore, their weakness in the lager datasets can be beneficial in the smaller datasets.

In contrast, our algorithm is dynamic, meaning that, a medoid is updated in each iteration. After finding a way into the center of cluster, the medoid may not stand still and some movements in its similar group of objects are predictable, therefore, slight deviation from the center is resulted. This deviation from the center shows its downsides on the results of the smaller datasets more than the larger datasets. Therefore, RKM did not entirely outperform in the datasets Iris, zoo and wine. But, in the datasets S_3, A_2,

mfeat-fac and WBC, our algorithms has the merit of being both fast and accurate, and the increased number of the local optimum cannot hinder RKM in finding clusters as much as the two other algorithms.

The running time of our algorithm is significantly smaller for the larger datasets like S_3, A_2, WBC and mfeat-fac. To understand the running time of these algorithms we should consider some facts. First, the running time of the algorithm is related to the number of clusters, therefore, the increase in the number of clusters, derives the running times up. Second, KHM should compute the similarity between new centers and the objects in the dataset, through iterations continuously. Therefore, its computational time is dependent to the computational time of the used similarity metric. In the case of the Euclidean distance that the running time has a linear dependency to dimension of the objects, the computational time of KHM, in each iteration is $O(k \times N \times D)$ as we have seen before. Thus, 216-dimension objects of mfeat-fac are the main reason for the long computational time of KHM. Third, *Simple and Fast k-medoids* selects a new medoid from a set of candidates which are the members of the same cluster. It updates the medoids in each iteration at the cost of $O(N)$ and therefore it finds the clusters faster than KHM. But, in our implementation the increase in the number of members of clusters makes the *Simple and Fast k-medoids* slower than KHM as we can see in the results of datasets S_3 and WBC. Our proposed algorithm updates the medoids in each iteration at the cost of $O(k \times m)$ where m is the number of members in G. It is much faster than the two other algorithms for larger datasets but in our implementation the overheads of relocating the medoids in step 2.2 makes it slower than *Simple and Fast k-medoids* for the smaller datasets.

As we mentioned before, the real dataset was widely used in many research and comparing the result with other algorithms is possible by using external measures. For example, the superiority of our results is revealed by reviewing the results of other algorithms like *K*-means, PSOKHM, ant clustering with KHM, ant colony clustering, data swarm clustering [20,25,41].

5. Conclusion

In this paper we introduced a novel *k*-medoids algorithm that uses a useful ranking function for finding the centrally located objects. We clarified the definition of local optimum for partitioning approach of clustering and showed that the algorithm can find all the Gaussian-shaped clusters if the right number of them is given. The initialization does not lead the algorithm into the local optimum. The similarity among objects in dataset is computed once and updating the medoids costs $O(k \times m)$ per iteration, where k is the number of clusters and m shows the number of members in a group needed to select the next medoids. To evaluate our algorithm, two artificial datasets and five real datasets were employed. *K*-Harmonic Means (KHM), our algorithm and *Simple and Fast k-medoids* compared with the help of four well-known external validation measures. The results revealed that KHM and *Simple and Fast k-medoids* suffer from local optimum. Our algorithm found the clusters in larger datasets faster and more accurate than the two other algorithms and it seems that ranked *k*-medoid clustering is a suitable algorithm for large dataset.

Acknowledgement

This study was part of a M.S. thesis supported by Tehran University of Medical Sciences (Grant No.: 541).

Appendix A. Supplementary material

Supplementary data associated with this article can be found, in the online version, at http://dx.doi.org/10.1016/j.knosys.2012.10.012.

References

[1] J. Han, M. Kamber, J. Pei, Data Mining: Concepts and Techniques, Morgan Kaufmann, 2006.
[2] P.C. Chang, C.H. Liu, C.Y. Fan, Data clustering and fuzzy neural network for sales forecasting: a case study in printed circuit board industry, Knowledge-Based Systems 22 (5) (2009) 344–355.
[3] E. Hadavandi, H. Shavandi, A. Ghanbari, Integration of genetic fuzzy systems and artificial neural networks for stock price forecasting, Knowledge-Based Systems 23 (8) (2010) 800–808.
[4] V. Subramanyam Rallabandi, S. Sett, Knowledge-based image retrieval system, Knowledge-Based Systems 21 (2) (2008) 89–100.
[5] M. ElAlami, Supporting image retrieval framework with rule base system, Knowledge-Based Systems 24 (2) (2011) 331–340.
[6] J.D. Martin-Guerrero, A. Palomares, E. Balaguer-Ballester, E. Soria-Olivas, J. Gomez-Sanchis, A. Soriano-Asensi, Studying the feasibility of a recommender in a citizen web portal based on user modeling and clustering algorithms, Expert Systems with Applications 30 (2) (2006) 299–312.
[7] J. Ponomarenko, T. Merkulova, G. Orlova, O. Fokin, E. Gorshkov, M. Ponomarenko, Mining DNA sequences to predict sites which mutations cause genetic diseases, Knowledge-Based Systems 15 (4) (2002) 225–233.
[8] J. Shi, Z. Luo, Nonlinear dimensionality reduction of gene expression data for visualization and clustering analysis of cancer tissue samples, Computers in Biology and Medicine 40 (8) (2010) 723–732.
[9] D. Sebiskveradze, V. Vrabie, C. Gobinet, A. Durlach, P. Bernard, E. Ly, M. Manfait, P. Jeannesson, O. Piot, Automation of an algorithm based on fuzzy clustering for analyzing tumoral heterogeneity in human skin carcinoma tissue sections, Laboratory Investigation 91 (5) (2011) 799–811.
[10] S. Kalyani, K. Swarup, Particle swarm optimization based *K*-means clustering approach for security assessment in power systems, Expert Systems with Applications 38 (9) (2011) 10839–10846.
[11] T. Zhang, R. Ramakrishnan, M. Livny, BIRCH: an efficient data clustering method for very large databases, in: ACM SIGMOD Conf. Management of Data, 1996.
[12] G. Karypis, E.H. Han, V. Kumar, Chameleon: hierarchical clustering using dynamic modeling, Computer 32 (8) (1999) 68–75.
[13] L. Kaufman, P.J. Rousseeuw, Finding Groups in Data: An Introduction to Cluster Analysis, vol. 39, Wiley Online Library, 1990.
[14] J. Sander, M. Ester, H.P. Kriegel, X. Xu, Density-based clustering in spatial databases: the algorithm GDBSCAN and its applications, Data Mining and Knowledge Discovery 2 (2) (1998) 169–194.
[15] C. Fraley, A.E. Raftery, Model-based clustering, discriminant analysis, and density estimation, Journal of the American Statistical Association 97 (458) (2002) 611–631.
[16] J. Vesanto, E. Alhoniemi, Clustering of the self-organizing map, IEEE Transactions on Neural Networks 11 (3) (2000) 586–600.
[17] W. Wang, J. Yang, R. Muntz, STING: a statistical information grid approach to spatial data mining, in: The International Conference on Very Large Databases, 1997.
[18] R. Xu, D. Wunsch, Survey of clustering algorithms, IEEE Transactions on Neural Networks 16 (3) (2005) 645–678.
[19] J. MacQueen, Some methods for classification and analysis of multivariate observations, in: The 5th Berkeley Symposium Math, Statistic and Probability, Berkeley, CA, 1967.
[20] F. Yang, T. Sun, C. Zhang, An efficient hybrid data clustering method based on *K*-harmonic means and particle swarm optimization, Expert Systems with Applications 36 (6) (2009) 9847–9852.
[21] J.C. Bezdek, R. Ehrlich, FCM: the fuzzy c-means clustering algorithm, Computers and Geosciences 10 (2–3) (1984) 191–203.
[22] B. Zhang, M. Hsu, U. Dayal, *K*-harmonic means – a data clustering algorithm, Hewllet-Packard Research Laboratory Technical Report HPL-1999-124, 1999.
[23] U. Maulik, S. Bandyopadhyay, Genetic algorithm-based clustering technique, Pattern recognition 33 (9) (2000) 1455–1465.
[24] Y.T. Kao, E. Zahara, I. Kao, A hybridized approach to data clustering, Expert Systems with Applications 34 (3) (2008) 1754–1762.
[25] H. Jiang, S. Yi, J. Li, F. Yang, X. Hu, Ant clustering algorithm with *K*-harmonic means clustering, Expert Systems with Applications 37 (12) (2010) 8679–8684.
[26] J. Senthilnath, S. Omkar, V. Mani, Clustering using firefly algorithm: performance study, Swarm and Evolutionary Computation 1 (3) (2011) 164–171.
[27] C. Zhang, D. Ouyang, J. Ning, An artificial bee colony approach for clustering, Expert Systems with Applications 37 (7) (2010) 4761–4767.
[28] K.E. Parsopoulos, M.N. Vrahatis, Particle Swarm Optimization and Intelligence: Advances and Applications, Information Science Reference, New York, 2010.

[29] H.S. Park, C.H. Jun, A simple and fast algorithm for *K*-medoids clustering, Expert Systems with Applications 36 (2) (2009) 3336–3341.
[30] Q. Zhang, I. Couloigner, A new and efficient *K*-medoid algorithm for spatial clustering, Computational Science and Its Applications_ICCSA 2005 (2005) 207–224.
[31] R.T. Ng, J. Han, Clarans: a method for clustering objects for spatial data mining, IEEE Transactions on Knowledge and Data Engineering 14 (5) (2002) 1003–1016.
[32] Y. Tian, D. Liu, H. Qi, *K*-harmonic means data clustering with differential evolution, in: International Conference on Future BioMedical Information Engineering, 2009.
[33] Clustering Datasets. <http://cs.joensuu.fi/sipu/datasets> (retrieved 01.08.12).
[34] P. Franti, O. Virmajoki, Iterative shrinking method for clustering problems, Pattern Recognition 39 (5) (2006) 761–775.
[35] UCI Machine Learning Repository, 2010. <http://archive.ics.uci.edu/ml/datasets.html> (retrieved 01.08.12).
[36] R.M. Aliguliyev, Performance evaluation of density-based clustering methods, Information Sciences 179 (20) (2009) 3583–3602.
[37] B.G. Mirkin, Mathematical Classification and Clustering, vol. 11, Springer, 1996.
[38] J. Wu, J. Chen, H. Xiong, M. Xie, External validation measures for *K*-means clustering: a data distribution perspective, Expert Systems with Applications 36 (3) (2009) 6050–6061.
[39] D. Steinley, Properties of the Hubert–Arable adjusted rand index, Psychological Methods 9 (3) (2004) 386.
[40] K.Y. Yeung, W.L. Ruzzo, Details of the adjusted rand index and clustering algorithms, supplement to the paper "An empirical study on principal component analysis for clustering gene expression data", Bioinformatics 17 (9) (2001) 763–774.
[41] C. Veenhuis, M. Köppen, Data swarm clustering, Swarm Intelligence in Data Mining 34 (2006) 221–241.