Pattern Recognition 79 (2018) 402–413

Contents lists available at ScienceDirect

Pattern Recognition

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

I-k-means−+: An iterative clustering algorithm based on an enhanced version of the *k*-means

Hassan Ismkhan

Department of Computer Engineering, University of Bonab, Bonab, East Azerbaijan, Iran

ARTICLE INFO

Article history:
Received 27 May 2017
Revised 31 December 2017
Accepted 11 February 2018
Available online 17 February 2018

Keywords:
k-means
Solution improving
Accurate *k*-means
Iterative improvement

ABSTRACT

The *k*-means tries to minimize the sum of the squared Euclidean distance from the mean (SSEDM) of each cluster as its objective function. Although this algorithm is effective, it is too sensitive to initial centers. So, many approaches in the literature have focused on determining suitable initial centers. However, selecting suitable initial centers is not always possible, especially when the number of clusters is increased. This paper proposes an iterative approach to improve quality of the solution produced by the *k*-means. This approach tries to iteratively improve the quality of solution of the *k*-means by removing one cluster (minus), dividing another one (plus), and applying re-clustering again, in each iteration. This method called iterative k-means minus–plus (I-k-means−+). The I-k-means−+ is speeded up using some methods to determine which cluster should be removed, which one should be divided, and how to accelerate the re-clustering process. Results of experiments show that I-k-means−+ can outperform k-means++, to be known one of the accurate version of the *k*-means, in terms of minimizing SSEDM. For some instances, the accuracy of I-k-means−+ is about 2 times higher than both the *k*-means and k-means++, while it is faster than k-means++, and has the reasonable runtime, in comparison with the *k*-means.

1. Introduction

Data clustering covers wide range of applications like video processing [1], image processing [2–4], text analysis [5,6], bioinformatics [7,8], wireless sensor networks [9], analysis for web social networks [10], and document clustering [11,12].

In order to define data clustering, let $I=\{x_1, x_2 \ldots x_n\}$ be an instance with n number of elements, and each element is characterized by m attributes: $x_i=\{x_{i,1}, x_{i,2}, \ldots x_{i,m}\}$. The aim of clustering is to group these elements into the k subsets such that pairs of subsets do not have any common element, union of subsets is equal to I, and elements in the same subset (called a cluster) are more similar to each other than to those in other groups.

Many clustering algorithms like DBSCAN [13], CURE [14], Chameleon [15], and k-MS [16] have been proposed in the literature. The *k*-means clustering algorithm [17] is one of the most influential data mining algorithms [18], which tries to minimize the sum of squared Euclidean distances to the mean of each group (SSEDM) as its objective function.

For an input I, and solution S, where $S=S_1 \cup S_2 \cup \ldots \cup S_k$, and $S_i \cap S_j=\emptyset$ $(i \neq j)$, the SSEDM value of S is calculated as:

$$SSEDM(S) = SSEDM\ (I,\ S,\ k)\ =\ \sum_{i\ =1}^{k} SSEDM(S_i) \tag{1}$$

In this paper, $SSEDM(S_i)$ is called partial SSEDM of cluster S_i, and can be computed using following equation:

$$SSEDM(S_i) = \sum_{\forall P \in S_i} \mathrm{dis}(P,\ mean(S_i))^2 \tag{2}$$

For Eq. (2), $dis(,)$ indicates the Euclidian distance between two points, and $mean(S_i)$ points to the center of cluster S_i.

Exactly minimizing the SSEDM, even for two dimensional (number of attributes for each data point is two) instances, is an *NP*-Hard optimization problem [19]. However, for instances with the linearly separable clusters, the *k*-means can effectively minimize SSEDM [20]. To minimize the SSEDM, and obtaining k number of clusters, the *k*-means operates instructions as follows: (1) select k number of points as initial centers, (2) assign each data point to its nearest center, (3) update location of centers, (4) repeat instructions 2 and 3 until convergence. Although the *k*-means is an effective algorithm, its performance, in terms of obtained SSEDM, strongly depends on the first instruction, where the initial centers are selected. It should be considered that good initial centers can also increase the speed of the convergence of the *k*-means. Therefore, many initialization methods are proposed for the *k*-means [21–23]. A review of some of these methods can be found in [24]. Although using good initialization method can increase the accuracy of *k*-means, even with utilizing these techniques, the accuracy of the *k*-means may not be satisfactory, especially, when the number of required clusters, k, is large. Fig. 1(B) demonstrates a solution obtained by the k-means++ [21] for a synthetic instance

E-mail addresses: esmkhan@gmail.com, H.Ismkhan@bonabu.ac.ir

https://doi.org/10.1016/j.patcog.2018.02.015
0031-3203/

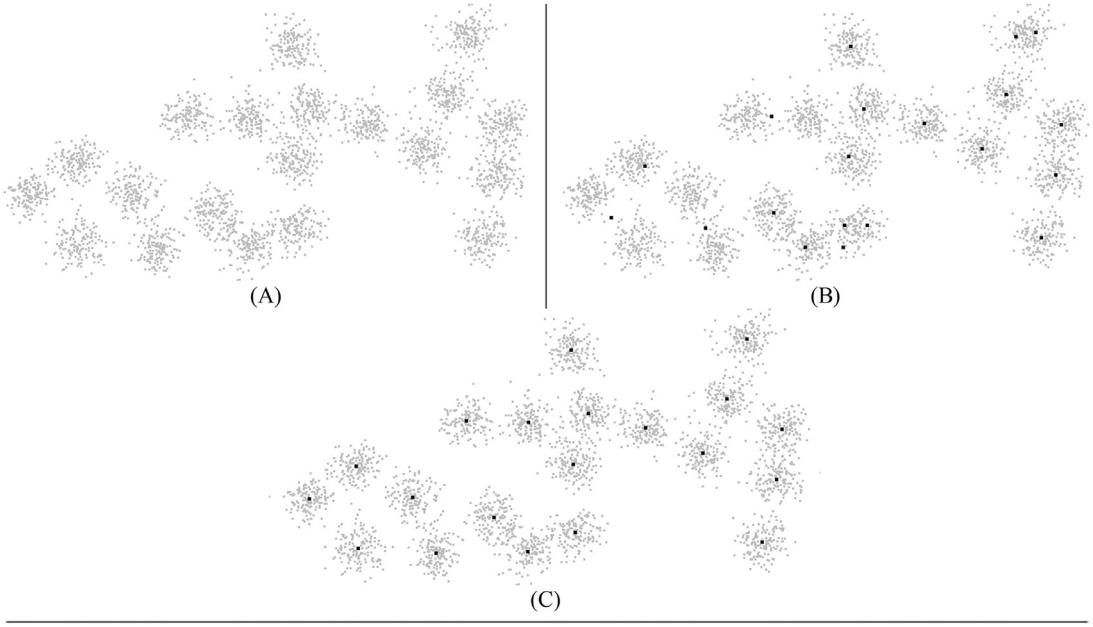

Fig. 1. (A) Original dataset with 20 clusters. (B) A solution obtained by k-means++, with SSEDM = 23829449829. (C) A solution with SSEDM = 12146257522. Perhaps it is the possible optimal solution.

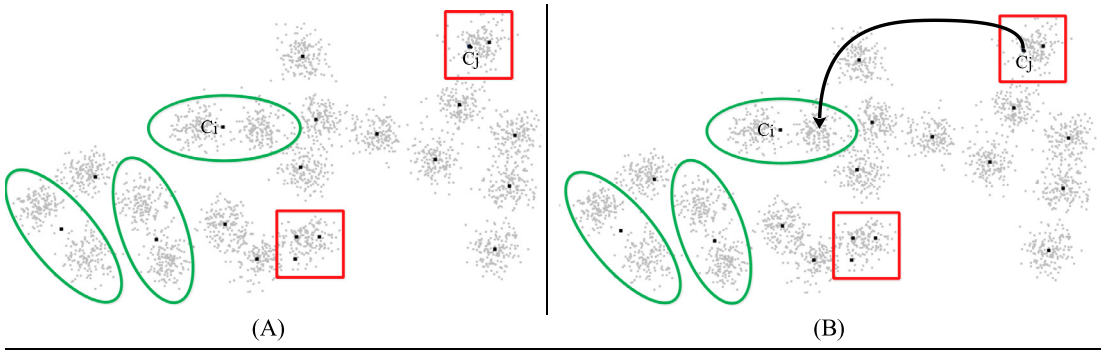

Fig. 2. (A) Each oval includes a cluster which can be split, and each square includes at least one cluster which can be removed. (B) Changing the location of C_j into cluster C_i, and applying re-clustering can improve the solution.

[25], which is shown in Fig. 1(A). Although the k-means++ utilizes one of the most accurate initialization methods, the SSEDM of its obtained solution is too larger than a solution shown in Fig. 1(C).

The example in Fig. 1 indicates that even after utilizing a good initialization method, the solution may need a process to be improved. Fig. 2 shows how solution obtained by the k-means++ can be improved. Each oval in Fig. 2(A) includes a cluster which can be split into two ones, where each square includes clusters that one or more can be removed. It should be considered that when a cluster is removed, its data points are redistributed among other suitable clusters. In Fig. 2(A), we can improve the quality of solution by removing cluster with center C_j and splitting cluster with center C_i. Let us call this, minus-plus phase, in which an existing cluster is removed and another existing cluster is split. In Fig. 2, if C_i is center of the cluster to be split, and C_j is center of the cluster to be removed, then minus-plus phase can perform by changing the location of center C_j into a data point in cluster with center C_i, and applying the *k*-means, again.

This paper proposes an iterative method, which improves the solution produced by the *k*-means. The proposed method tries to iteratively apply minus-plus phase, so it is called I-k-means−+ (iterative k-means minus plus). In each iteration, I-k-means−+ tries to quickly find a suitable pair of clusters for applying minus-plus phase, using some heuristics. Results of performed experiments appear that generally, the accuracy of I-k-means−+ is higher than other competitors, in terms of minimizing SSEDM. Furthermore, using the proposed heuristics in this paper, the speed of I-k-means−+ is also acceptable, in comparison with the *k*-means, where in some cases it is faster than the *k*-means. In addition to these advantages, perhaps, this idea can be also applicable to algorithms with similar objective function, like fuzzy c-means (FCM) clustering algorithm, which is left as future works.

Therefore, the next section reviews the *k*-means, Section 3 presents the proposed algorithm, Section 4 demonstrates the results of performed experiments, and finally, the last section summarizes this paper.

2. Recent works for the *k*-means

As the *k*-means is too sensitive to initial centers, many research works try to propose an effective method to initialize centers of the *k*-means. Therefore, a part of this section is devoted to reviewing some of these methods. In addition, the *k*-means loses its speed, as it requires large computational effort in assigning step, where each data point should be assigned to its nearest center among existing centers. So, the second part of this section reviews some of the important methods which try to speed up the *k*-means.

2.1. Initialization methods for the k-means

The Maxmin [26,27] selects a random data point as the first center, then to select the ith center, takes a data point with the largest distance to its nearest previously selected center.

Paper [23] divides space of dataset into some buckets, using kd-trees, then determines the density of each bucket. Each bucket is denoted by a point which is means of points located in that bucket. A bucket with the largest value of density is selected as the first center. The mean point of a bucket, which has the maximum value of multiplication of its density by the distance from its nearest center, is selected as the ith center.

The k-means++ [21], chooses the first center, among data points of data set, randomly. It chooses a data point $x \in X$ as the ith center with probability $\frac{D(X)^2}{\sum_{x\in X} D(X)^2}$, where $D(x)$ denotes the shortest distance from a data point to the closest center we have already chosen.

The PCA-Part [28] starts with an initial cluster that contains all data points, then completes the process in $k - 1$ steps, where, in each step, takes a cluster with the largest partial SSEDM, and divides it into two separate sub-clusters. To divide a cluster, it uses a hyperplane that passes through the cluster centroid and is orthogonal to the direction of the principal eigenvector of the covariance matrix.

The global k-means (GKM) [29] starts with two centers and adds a center at each time. To add the ith center, GKM considers each data point as a candidate for the ith center. To choose the ith center, the data point which leads to the minimum value of $b_n = \sum_{j=1}^{n} \max\{0,\ d_{i-1}^{j} - ||x_n - x_j||^2\}$ is selected as the ith center. Modified GKM (MGKM) [30], to select the starting point of the ith center, proposes a different way in which minimizes another auxiliary cluster function. Performed experiments in [30] show that the MGKM can minimize the SSEDM better than the GKM, but it is slower than the GKM. Paper [31] tries to speedup GKM by making use of the cluster membership and geometrical information of a data point. The fast MGKM (FMGKM) [32] exploits information gathered in previous iterations of the incremental algorithm and decreases memory usage and increases speed of the MGKM.

To tackle initialization problem of the *k-means*, the MinMax version of k-means [33] changes objective function, and instead using SSEDM, utilizes *maximum intra-cluster variance* ($\varepsilon_{\max}$) as a potential objective function to be minimized. As the direct minimization of the $\varepsilon_{\max}$ is difficult, a relaxation of this function is considered to be minimized. In addition, the MinMax assigns a weight for each cluster, such that clusters with larger intra-cluster variance are allocated higher weights and these weights are learned automatically. Based on the claim in [33], the MinMax is less affected by initialization, and it can discover a high quality solution, even with bad initial centers.

The proposed mechanism by [22], to reduce computational time of selecting initial centers, tries to select two axes (dimensions) that best describe the change in the dataset at first. After this, in the process of selecting k centers, these two axes are used in all computations to calculate required distances between data points. Then, it determines the mean of data points according to selected axes. The farthest data point to the mean is selected as the first center. A data point with the maximum sum of distances from previous $i - 1$ centers, is selected as the ith center. This process is continued until all required centers to be selected.

2.2. Methods for speeding up the k-means

Reducing dimension of the dataset by using feature selection or feature extraction can speed up the k-means. Feature selection differs feature extraction, where, in feature selection a small subset of the input features is selected, while for feature extraction, a small set of new artificial features is constructed, based on input dataset. These mechanisms can considerably increase the speed of the k-means algorithm [34,35].

To speed up the k-means, paper [36] reduces computations of the inner loop of the k-means, where, in common version of the k-means, distances between each data point and centers are computed, and each data point is assigned to its closest center. Paper [36], speeds up this loop, by ignoring some distance computations of data points from some centers, using triangular inequality, and keeping some extra information, especially including an upper bound and k number of lower bounds for each data point. Paper [37] proposes an algorithm to reduce the computations of the inner loop, but its memory complexity is improved, and instead k number of lower bounds, it keeps only one lower bound for each data point.

The GAD [38,39] reduces computation cost of assigning data points to their nearest centers using the concepts of active and static centers. If the location of a center is changed for the next iteration, it is called active, otherwise, it is called static. To describe this method, please consider a case: for the next iteration, the nearest center of a data point becomes closer to the data point, so the static centers can be pruned in searching for the nearest center of the data point. The GAD considers such cases and tries to speed up the inner loop of the k-means by ignoring some centers in the process of searching the nearest centers of data points. The proposed algorithm in [40] is a type of filtering algorithms which makes use of center variations to speed up clustering process. It divides clusters into static and active groups, then it reduces the computational complexity by finding candidates for each node mainly from the set of active cluster centers.

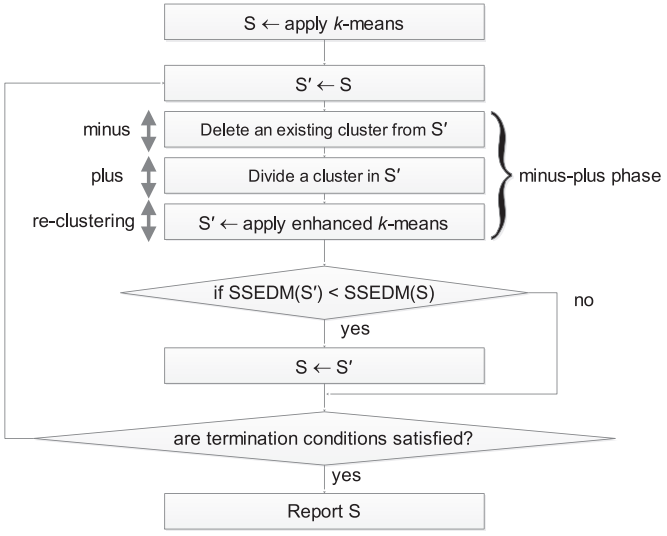

Fig. 3. A view of the I-k-means−+.

3. The I-k-means−+

The I-k-means−+ produces a solution by applying the k-means using initialization method in [41], then it starts an iterative process to improve quality of the produced solution, in terms of minimizing SSEDM. In each iteration, it removes a cluster (minus), divides another cluster into two ones (plus), and applies re-clustering algorithm using an enhanced version of the k-means. Fig. 3 shows a general view of the I-k-means−+. Fig. 4 shows how minus-plus phase (the process of removing a cluster, adding

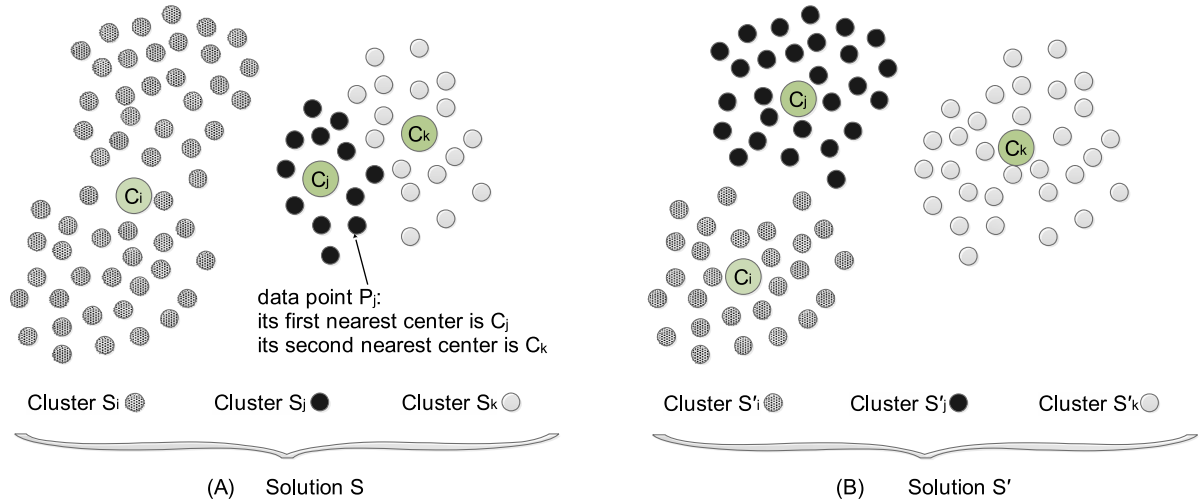

Fig. 4. A cluster S_i is divided into two clusters and a cluster S_j is removed, and a new solution S' is produced. It is obvious that when a cluster is removed (S_j), its data points are redistributed among other suitable clusters (S'_k). These are performed, simply, by changing C_j into a random data point in S_i and then applying re-clustering process.

another one and applying re-clustering algorithm) can improve the result, where a cluster S_i of solution S is divided into two clusters, and a cluster S_j is removed from S, then the improved cluster solution, S', is produced. The minus-plus can be done by changing the location of C_j as same as a random point in S_i and applying the k-means. However, there are two main challenges:

1 How we can find a suitable pair of clusters, like S_i and S_j. Considering all pairs is not reasonable, and would be too time-consuming.
2 How we can accelerate re-clustering process using the k-means, where only a local part of solution is needed to be re-clustered.

The rest of this section describes how up-coming challenges can be resolved, using some strategies to detect suitable pairs, and speeding up the re-clustering process by an enhanced version of the k-means. The last sub-section explain the I-k-means−+ by detailed instructions.

3.1. Detecting suitable pairs

Suitable pairs of clusters should be detected before applying minus-plus phase, because trial and error strategy of multiple performing minus-plus phase, especially its re-clustering process, would be too time-consuming. Therefore, before dividing cluster S_i and removing S_j, it should be determined or estimated what is added to SSEDM by removing S_j, which is the cost of removing S_j, and what is taken from SSEDM by dividing S_i, which is gain of dividing S_i.

Definition 1. For a solution S, the cost of removing a cluster S_j, denoted by *Cost(S_j)*, is what is added to the *SSEDM(S)*, when S_j is removed from S.

The ultimate amount of *Cost(S_j)* is the subtraction of what is added to *SSEDM(S)* from what is taken from *SSEDM(S)*. The value which is taken from *SSEDM(S)*, is the current value of *SSEDM(S_i)*, before removing S_i. We can approximately determine what value is added to *SSEDM(S)* by computing sum of the distance of each data point in S_j from its second nearest center. Therefore, *Cost(S_j)* can be estimated by the following equation:

$$Cost\left(S_j\right) = SSEDM\left(S_j\right) - \sum_{\forall P \in S_j} dis(P,\ CC_P)^2 \tag{3}$$

where CC_P is the second nearest center of P. It should be noted that the second nearest center can be determined in assigning phase of the k-means, without imposing additional time-cost.

Definition 2. For a solution S, the gain of dividing a cluster S_i, denoted by *Gain(S_i)*, is what value is decreased from the *SSEDM(S)* by dividing S_i from S to two clusters.

When a cluster like S_i is divided to two clusters S'_i and S'_j, the current value of *SSEDM(S_i)* is subtracted from *SSEDM(S)*, and versus, the sum of *SSEDM(S'_i)* and *SSEDM(S'_j)* are added to *SSEDM(S)*. So, *Gain(S_i)* can be computed by subtraction of the sum of *SSEDM(S'_i)* and *SSEDM(S'_j)* from the current value of *SSEDM(S_i)*:

$$Gain(S_i) = SSEDM(S_i) - \left(SSEDM\left(S'_i\right) + SSEDM\left(S'_j\right)\right) \tag{4}$$

For this equation the current value of *SSEDM(S_i)* is available, but *SSEDM(S'_i)* and *SSEDM(S'_j)* are not available, an estimation of these values is needed. To compute these values approximately, we use current average distances of data points in S_i form its center, C_i:

$$X = \frac{1}{|S_i|}\sum_{\forall P \in S_i} dis(P, C_i) \tag{5}$$

Where C_i is the center of cluster S_i which is to be divided, and $|S_i|$ is the number of data points belongs to S_i. In fact, X denotes average distance of each data point in S_i from C_i.

Now, for the both produced clusters, we estimate the distance of each data point from its center by $X/2$, and the number of data points belongs to each produced cluster by $|S_i|/2$, so, we have:

$$SSEDM\left(S'_i\right) \approx SSEDM\left(S'_j\right) \approx \frac{|S_i|}{2} \times \left(\frac{X}{2}\right)^2 \tag{6}$$

If we use following estimation:

$$SSEDM(S_i) \approx |S_i| \times X^2 \tag{7}$$

Using (4), (6) and (7), we have:

$$\begin{aligned} Gain(S_i) &\approx |S_i| \times X^2 - 2\ \times \frac{|S_i|}{2} \times \left(\frac{X}{2}\right)^2 \\ &= \frac{3}{4}|S_i| \times X^2 \approx \frac{3}{4} SSEDM(S_i) \end{aligned} \tag{8}$$

Then, by coefficient $\alpha = \frac{3}{4}$, we can estimate gain of dividing a cluster S_i, by following equation:

$$Gain(S_i) \approx \alpha SSEDM(S_i) \tag{9}$$

Heuristic 1. A pair of clusters, S_i and S_j can be considered for the minus-plus phase, S_i for dividing, and S_j for removing, if *Gain(S_i) > Cost(S_j)*.

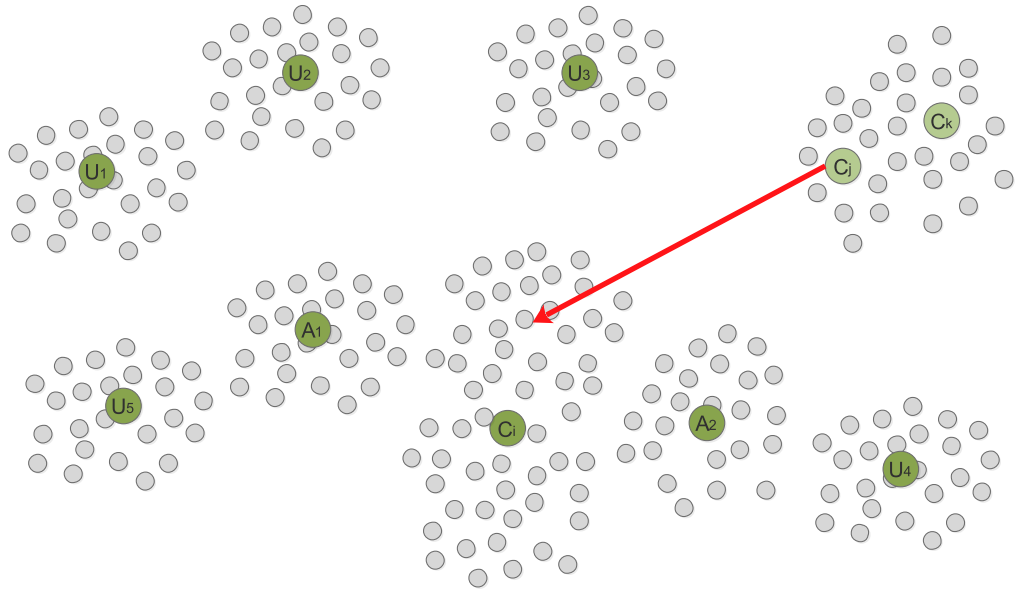

Fig. 5. When only one center C_j of a clustering solution is updated, then in order to accelerate, re-clustering can be applied topically.

Aside from amounts of gains and costs, there is a concept which can be utilized to select suitable pairs. In Fig. 4, after applying minus-plus phase, and observing $SSEDM(S') < SSEDM(S)$, it is not reasonable to divide S_k, which was an adjacent of a removed cluster, in the rest of the iterative process. In addition, removing both S_i and S_j cannot be reasonable, where they have already been produced by dividing a cluster. Before stating these rules in the form of heuristics, two definitions are needed.

Definition 3. A cluster S_j with center C_j is an adjacent of a cluster S_i with center C_i if and only if, there is a data point P in S_i (its first nearest center is C_i) such that the second nearest center of P is C_j.

When a cluster S_j is adjacent to a cluster S_i, it does not imply S_i is adjacent to S_j.

Definition 4. A cluster S_j is a strong adjacent to a cluster S_i if and only if S_j is an adjacent of S_i, and S_i is an adjacent of S_j.

For example, in Fig. 4(A), S_j is a strong adjacent of S_k, and S_j is an adjacent of S_i, but S_i is not an adjacent S_j.

Heuristic 2. If a cluster is produced by dividing another cluster in the current minus-plus phase, it and its strong adjacent clusters cannot be removed in the next iterations.

Heuristic 3. If a cluster is removed in the current minus-plus phase, its strong adjacent clusters cannot be divided in the next iterations.

3.2. *The topical* k-*means*

The topical *k*-means (t-k-means) is an enhanced version of the *k*-means which is only applied to potentially affected points. In the minus-plus phase of the I-k-means−+, only a part of the solution, which has been produced in one of the previous iterations, is changed, so as global re-clustering of the instance is time-consuming, the t-k-means is only applied topically. For example, in Fig. 5, I-k-means−+, in minus-plus phase, decides that the cluster with center C_j should be removed, and the cluster with C_i should be divided, therefore it changes the location of C_j as a random point in the cluster with center C_i. The t-k-means accelerates the re-clustering process by considering affected points and ignoring unaffected points. After following definition, which describes affected clusters and points, we introduce the t-k-means in more details.

Definition 5. In a clustering solution S, a data point P is affected point of a cluster center C_i if and only if the first or the second nearest center of P is C_i.

The following instructions show t-k-means which are applied to a solution in which only the center C_j of a cluster S_j is changed as a random point in S_i with the center C_i.

Step#1.

1. Let *AC* (active centers) be an initially empty set of centers.
2. $AC \leftarrow AC \cup \{C_i\} \cup \{C_j\}$.
3. *AC-Adjacent* ←adjacent center of C_j before changing.
4. *AP* ← affected points of C_j before changing.

Step#2.

1. If *AC* is an empty set go to the end.
2. *AC-Adjacent* ← adjacent centers of centers in *AC*.
3. *Potential-AC* ← {}.
4. AP ← affected points of centers in *AC*.

Step#3.
For each affected point, *P*, of centers in *AC*

1. Update the first and the second nearest centers of *P*, considering $AC \cup$ *AC-Adjacent*.
2. If the first center of *P* is changed from C_x to C_y, then *Potential-AC* ← *Potential-AC* $\cup \{C_x\} \cup \{C_y\}$

Step#4. Update centers.
Step#5. *AC* ← *Potential-AC*
Step#6. Go to Step#2.
Step#7. The end of the algorithm.

The t-k-means, for solution *S*, in Fig. 5, determines $AC = \{C_i, C_j\}$, *AC-Adjacent* $= \{C_k, A_1, A_2\}$ and affected points of centers in *AC* and affected points of C_j before changing, during Step#1 and Step#2. It should be considered, some points from clusters with centers C_k, C_j, C_i, A_1, A_2, and U_3 are included in *AP*. After first execution of Step#3, not only the first and the second nearest centers of points in *AP* are updated, we have *Potential-AC* $= \{C_k, C_j, C_i\}$. In Step#6, t-k-means goes to Step#2 again. This process is continued until in Step#2, *AC* is empty.

3.3. The I-k-means−+ via detailed instructions

The I-k-means−+ can be stated in the following instructions.

Instruction#1. Apply the *k*-means using initialization method in [41], and produce solution S with clusters S_i $(1 \le i \le k)$.

Instruction#2. Consider a variable *#success* ← *0*.

Instruction#3. Among clusters with the following condition, select a cluster S_i with the largest value of *Gain*. If there is not such a cluster go the end of the algorithm.

1. The cluster S_i should not be marked as an *indivisible* cluster.

Instruction#4.

If there are *k/2* clusters have a gain larger than S_i, then go to end. In fact, in this case, there are *k/2* clusters have larger SSEDM, so when these clusters are indivisible, therefore dividing S_i cannot be reasonable.

Instruction#5. Among clusters with following conditions, select a cluster S_j with the smallest value of *Cost*. If there is not such a cluster go the end of the algorithm.

1. $S_j \neq S_i$
2. $\mathrm{Cost}(S_j) < \mathrm{Gain}(S_i)$.
3. Pair S_i and S_j should not be marked as an unmatchable pair.
4. S_i is not an adjacent of S_j, and S_j is not an adjacent of S_i.
5. The cluster S_j should not be marked as an irremovable cluster.

Instruction#6.

1. If there are *k/2* clusters have cost smaller than S_j and satisfy conditions (1), (2), (3), (4) and (5), then go to end.
2. If there are *k/2* clusters have cost smaller than S_j, then S_i cannot be paired with any other cluster, so mark S_i as an *indivisible* cluster and go to Instruction#3 (consider another cluster to divide).

Instruction#7.

1. Save the current solution: $S' \leftarrow S$.
2. For S', change coordination of center of cluster S_j, as a random data point from a cluster S_i.
3. Update S' by applying the t-k-means

Instruction#8.

If $SSEDM(I, k, S') > SSEDM(I, k, S)$ then: the solution is not improved and the pair S_i and S_j should be marked as an *unmatchable* pair.

Otherwise if $SSEDM(I, k, S') < SSEDM(I, k, S)$ then:

1. Mark both S_i and S_j as an *irremovable* cluster.
2. Mark previous strong adjacent clusters of S_j, as an *indivisible* cluster.
3. $S \leftarrow S'$.
4. Mark the current strong adjacent clusters of S_i and S_j, as an *irremovable* cluster.
5. *#success* ← *#success* + *1*.

Instruction#9. If *#success* > *k* / 2, then go to end.

Instruction#10. The end of the algorithm.

4. Results of performed experiments

This section reports results of experiments. Three types of datasets are used in experiments. Datasets of the first type are two-dimensional synthetic problems [25], which let us show the accuracy of algorithms visually, where, an algorithm with a high-quality solution can be determined by observing the ultimate location of the centers. The second and the third types are real-world problems. Datasets of the second type are available via UCI machine learning repository. The third type contains a protein dataset, KDDCUP04Bio, which is rather large and comes from bioinformatics. Table 1 shows the detail of utilized datasets.

Table 1
Datasets used in experiments.

Type	Instance name		Dimension	#data points	#clusters
#1	A-series	A1	2	3000	20
		A2	2	5250	35
		A3	2	7500	50
	S-series	S1	2	5000	15
		S2	2	5000	15
		S3	2	5000	15
		S4	2	5000	15
		Birch1	2	100,000	100
#2	Iris		4	150	3
	Human-Activity-Recognition (HAR)		561	10,299	6
	ISOLET		617	7797	26
	Letter-Recognition (LR)		16	20,000	9
	Musk		168	6598	2
	Statlog (Shuttle)		9	58,000	7
#3	KDDCUP04Bio		74	145,751	2000

To evaluate the proposed I-k-means−+ (IKM−+), two other competitors including the *k*-means (KM), and k-means++ (KM++) participate in experiments as competitors of IKM−+. For KM, initial centers are chosen randomly, according to the second method of MacQueen [17], in which k random points from the dataset are chosen as initial centers. All algorithms are programmed using C++, and details of the machine are Intel® CoreTM i3-4150 CPU @ 3.50 GHz with 4 GB of random access memory. The SSEDM, as criteria for accuracy, and runtime of algorithms are reported for these algorithms. In addition, for each algorithm, maximum of partial SSEDMs [33] is also reported in the results. As Wilcoxon-Signed-Rank-test at a confidence level of 5% shows that difference between results of IKM−+ and each of other competitors are significant, so, no additional symbol is used in tables and figures for demonstrating results of this test.

4.1. Results on synthetic datasets

Results gained from 50 runs of participated algorithms on synthetic datasets are introduced in Table 2. For both maxima of partial SSEDMs and SSEDM, IKM−+ outperforms other algorithms, where its average runtime is also acceptable for each instance. Considering runtime for all instances, KM at most is 1.5 times better than IKM−+, where for some instances, IKM−+ is faster than KM. These results reveal that IKM−+ increases the accuracy of clustering without losing speed, so much.

The worst and the best SSEDM obtained by algorithms are not on the table, instead, we try to show them by visualizing. For A series, not only the best and average SSEDM gained by IKM−+ is better than KM and KM++ , the worst solution of IKM−+ is also better than the best solutions of KM and KM++. So, for A-series, the best solution of KM, the best solution of KM++, and versus, the worst solution of IKM−+ are shown. Figs. 6–8 show the best solution of KM, the best solution of KM++, and the worst solution of IKM−+ for A1, A2, and A3, respectively. Centers of the solutions found by algorithms in these diagrams, let us understand that for A-series, the worst solutions of IKM−+ are better than the best solutions of KM and KM++. For these two instances, the worst solutions of IKM−+ are similar to its best ones. For S-series, the best solutions gained by all three algorithms are similar. Like A-series, for S-series, the worst solutions of IKM−+ are similar to its best solutions. Fig. 9 shows the best solution of KM, the best solution of KM++ and the worst solution of IKM−+ for S1, S2, S3, and S4,

Table 2
Results of 50 runs on synthetic datasets.

	Maximum of partial SSEDMs			SSEDM			Runtime (s)			t_{IKM-+}/t_{KM}
	KM	KM++	IKM−+	KM	KM++	IKM−+	KM	KM++	IKM−+	
A1	5.39E + 09	4.08E + 09	**7.51E + 08**	2.08E + 10	1.73E + 10	**1.22E + 10**	**1.70E − 02**	2.10E − 02	2.50E − 02	1.45
A2	5.78E + 09	4.31E + 09	**7.12E + 08**	3.47E + 10	2.99E + 10	**2.03E + 10**	5.50E − 02	9.30E − 02	**5.00E − 02**	0.91
A3	7.58E + 09	4.98E + 09	**7.13E + 08**	5.23E + 10	4.29E + 10	**2.90E + 10**	1.17E − 01	2.10E − 01	**1.12E − 01**	0.96
S1	6.59E + 12	6.20E + 12	**8.54E + 11**	1.85E + 13	1.67E + 13	**8.92E + 12**	**2.00E − 02**	2.30E − 02	2.20E − 02	1.11
S2	5.78E + 12	5.25E + 12	**1.33E + 12**	2.01E + 13	1.82E + 13	**1.33E + 13**	2.10E − 02	2.70E − 02	**2.00E − 02**	0.93
S3	3.41E + 12	3.04E + 12	**1.56E + 12**	1.94E + 13	1.90E + 13	**1.69E + 13**	**2.70E − 02**	2.60E − 02	5.30E − 02	1.96
S4	2.56E + 12	2.19E + 12	**1.79E + 12**	1.70E + 13	1.67E + 13	**1.57E + 13**	**4.00E − 02**	4.20E − 02	4.00E − 02	1.00
Birch1	3.15E + 12	2.96E + 12	**1.05E + 12**	1.13E + 14	1.06E + 14	**9.28E + 13**	**1.81E + 01**	2.47E + 01	2.73E + 01	1.51

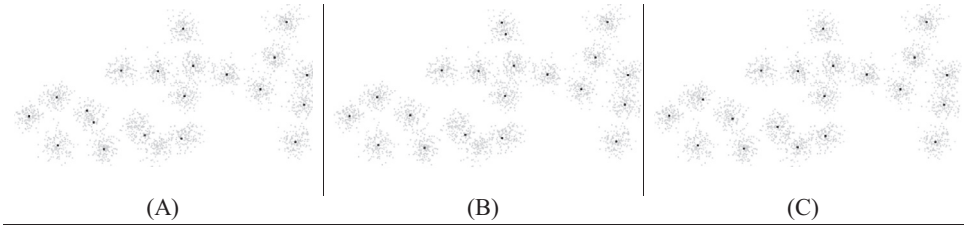

Fig. 6. For A1, the *best* solution of (A) KM, (B) KM++, and (C) the *worst* solution of IKM−+.

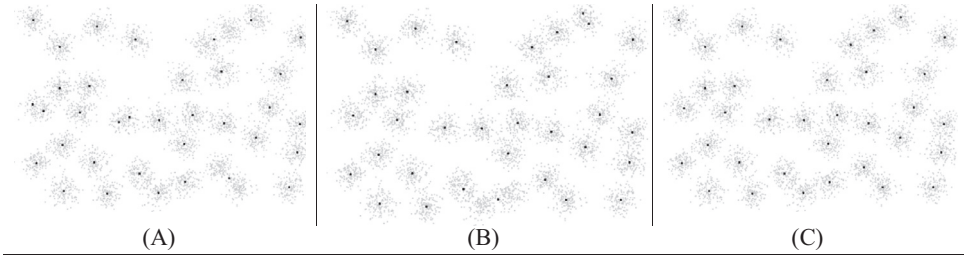

Fig. 7. For A2, the *best* solution of (A) KM, (B) KM++, and (C) the *worst* solution of IKM−+.

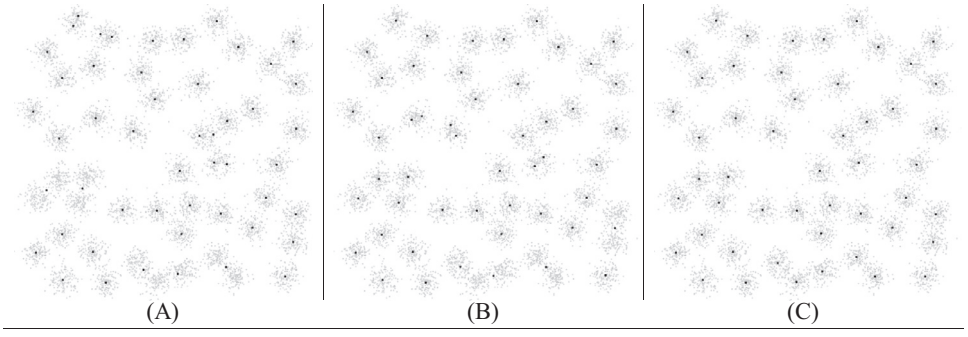

Fig. 8. For A3, the *best* solution of (A) KM, (B) KM++, and (C) the *worst* solution of IKM−+.

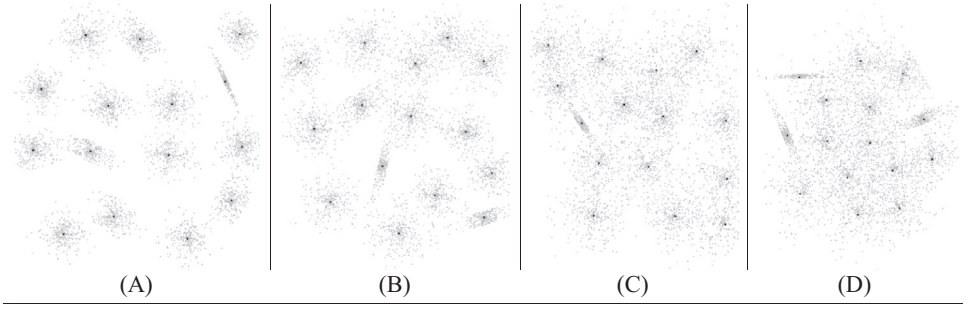

Fig. 9. For S1, S2, S3, and S4 the *best solution of KM, and KM++* and the *worst* solution of IKM−+ are similar to (A), (B), (C), and (D), respectively.

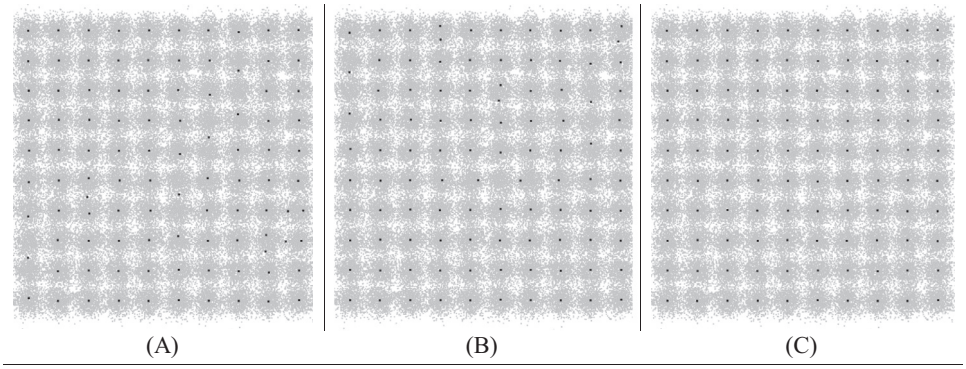

Fig. 10. For Birch1, the *best* solution of (A) KM, (B) KM++, and (C) the *worst* solution of IKM−+.

Table 3
Results on real-world datasets.

	Maximum of partial SSEDMs			SSEDM			Runtime (s)			t_{IKM-+}/t_{KM}
	KM	KM++	IKM−+	KM	KM++	IKM−+	KM	KM++	IKM−+	
Iris	6.53E+01	4.55E+01	**3.98E+01**	9.95E+01	8.41E+01	**7.89E+01**	0	0	0	1.00
HAR	5.02E+04	5.10E+04	**3.83E+04**	1.85E+05	1.85E+05	**1.82E+05**	1.696	1.635	4.641	2.74
ISOLET	4.23E+04	4.17E+04	**2.78E+04**	4.46E+05	4.47E+05	**4.41E+05**	5.066	6.712	6.207	1.23
LR	4.17E+04	4.29E+04	**3.93E+04**	6.20E+05	6.21E+05	**6.16E+05**	1.238	1.497	3.686	2.98
Musk	4.61E+09	4.47E+09	**4.35E+09**	6.09E+09	6.01E+09	**5.92E+09**	0.026	0.023	0.022	0.84
Statlog	**2.41E+08**	2.78E+08	3.32E+08	8.13E+08	7.99E+08	**4.35E+08**	0.426	0.434	0.342	0.80

Table 4
Effects of different settings of IKM−+ on SSEDM.

	SSEDM					
	v1	v2	v3	v4	v5	v6
Iris	7.89E+01	7.89E+01	7.89E+01	7.89E+01	9.69E+01	8.80E+01
HAR	1.82E+05	1.82E+05	1.82E+05	1.82E+05	1.82E+05	1.82E+05
ISOLET	4.41E+05	4.41E+05	4.40E+05	4.41E+05	4.40E+05	4.40E+05
LR	6.16E+05	6.16E+05	6.15E+05	6.16E+05	6.15E+05	6.16E+05
Musk	5.92E+09	5.92E+09	5.92E+09	5.92E+09	6.08E+09	6.10E+09
Statlog	4.35E+08	5.44E+08	4.35E+08	4.35E+08	5.70E+08	5.59E+08

Table 5
Effects of different settings of IKM−+ on runtime.

	Runtime (s)					
	v1	v2	v3	v4	v5	v6
Iris	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
HAR	5.55E+00	5.47E+00	5.55E+00	5.54E+00	3.42E+00	3.31E+00
ISOLET	7.30E+00	7.19E+00	7.53E+00	7.26E+00	1.00E+01	1.14E+01
LR	5.07E+00	4.20E+00	5.30E+00	4.76E+00	5.20E+00	5.55E+00
Musk	2.00E−02	2.00E−02	2.00E−02	2.00E−02	2.00E−02	2.00E−02
Statlog	3.90E−01	2.60E−01	4.10E−01	4.00E−01	5.50E−01	5.40E−01

respectively. Similar to A-series and S-series, for Birch1, as shown in Fig. 10, the worst solution obtained by IKM−+ is better than the best solutions of other two competitors.

4.2. Results on real-world datasets

Results of applying competitors to real-world datasets are shown in Table 3, which introduces the average of maximum of partial SSEDMs, the average SSEDM, and the average runtime of 50 runs of each competitor. The SSEDM values obtained by IKM−+ are better than the other competitors for all instances. In addition, for the maximum of partial SSEDMs, IKM−+ outperforms other competitors, except in the last instance, in which maximum of partial SSEDMs gained by KM is better than IKM−+'s one.

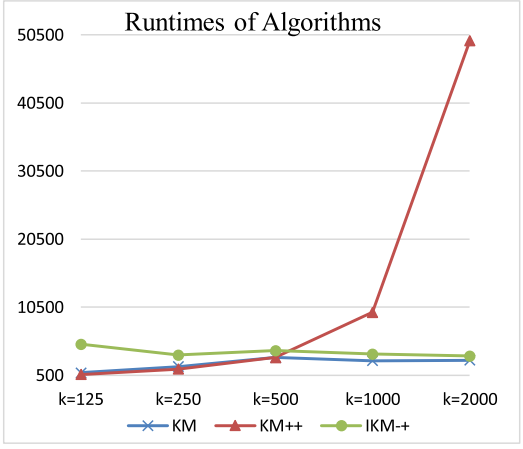

Fig. 11. Runtime of KM++ is dramatically increased by increasing number of clusters.

4.3. Effects of α and number of improvements on IKM−+

In Eq. (9), to compute Gain(S_i), we estimate coefficient $\alpha = 0.75$. Here, we have an experiment in which the effects of this coefficient are explored using alternative values $\alpha = 0.5$, and $\alpha = 1.0$. In addition, in the algorithm of the IKM−+, we limit the number of improvements to $k/2$, the half of the number of clusters, using instruction "If #success > $k/2$, then go to end". It should be noted that this number may be reasonable, where it is impossible that all centers of the produced solution by the k-means are been located in only one grand-truth cluster. However, in this experiment, we explore effects of the maximum number of improvements by k, instead of $k/2$. Moreover, we consider effects of different initialization methods such as random initialization and KM++ for the first use of the k-means in IKM−+. So, we considered five versions of IKM−+:

- v1: IKM−+ (original setting with $\alpha = 0.75$)
- v2: IKM−+ with $\alpha = 0.5$

Table 6
Results on KDDCUP04Bio.

	Maximum of partial SSEDMs			SSEDM			Runtime (s)			t_{IKM-+}/t_{KM}
	KM	KM++	IKM−+	KM	KM++	IKM−+	KM	KM++	IKM−+	
$k=125$	8.34E+10	1.70E+10	**1.58E+09**	2.30E+11	1.62E+11	**1.43E+11**	**851.35**	618.62	5041.59	5.92
$k=250$	8.09E+10	7.56E+09	**7.05E+08**	2.06E+11	1.31E+11	**1.21E+11**	**1771.26**	1389.27	3466.35	1.96
$k=500$	6.08E+10	1.69E+09	**3.47E+08**	1.71E+11	1.07E+11	**1.04E+11**	**3111.37**	3112.03	4096.47	1.32
$k=1000$	2.81E+10	7.60E+08	**2.11E+08**	1.26E+11	9.10E+10	**8.94E+10**	**2646.5**	9732.63	3624.23	1.37
$k=2000$	7.07E+09	3.82E+08	**1.72E+08**	9.21E+10	**7.71E+10**	7.73E+10	**2703.1**	49,635.6	3309.17	1.22

Table 7
Effects of increasing number of clusters up to 100 on HAR.

	SSEDM			$SSEDM_{IKM-+}$/		Runtime (s)		
	KM	KM++	IKM−+	$SSEDM_{KM}$	$SSEDM_{KM++}$	KM	KM++	IKM−+
$k=12$	1.63E+05	1.63E+05	1.62E+05	1.01	1.01	3.56E+00	4.08E+00	1.60E+01
$k=25$	1.45E+05	1.45E+05	1.44E+05	1.01	1.01	7.74E+00	9.35E+00	2.96E+01
$k=50$	1.30E+05	1.30E+05	1.29E+05	1.01	1	1.53E+01	2.08E+01	2.76E+01
$k=100$	1.16E+05	1.16E+05	1.15E+05	1.01	1.01	2.19E+01	5.01E+01	2.98E+01

Table 8
Effects of increasing number of clusters up to 100 on ISOLET.

	SSEDM			$SSEDM_{IKM-+}$/		Runtime (s)		
	KM	KM++	IKM−+	$SSEDM_{KM}$	$SSEDM_{KM++}$	KM	KM++	IKM−+
$k=12$	5.16E+05	5.16E+05	5.14E+05	1	1	1.99E+00	2.20E+00	5.47E+00
$k=25$	4.50E+05	4.49E+05	4.43E+05	1.02	1.01	4.36E+00	5.38E+00	5.77E+00
$k=50$	4.06E+05	4.05E+05	4.01E+05	1.01	1.01	8.99E+00	1.41E+01	1.29E+01
$k=100$	3.70E+05	3.69E+05	3.67E+05	1.01	1.01	1.53E+01	3.65E+01	1.28E+01

Table 9
Effects of increasing number of clusters up to 100 on LR.

	SSEDM			$SSEDM_{IKM-+}$/		Runtime (s)		
	KM	KM++	IKM−+	$SSEDM_{KM}$	$SSEDM_{KM++}$	KM	KM++	IKM−+
$k=12$	8.13E+05	8.10E+05	8.08E+05	1.01	1	5.27E−01	5.98E−01	1.16E+00
$k=25$	6.27E+05	6.28E+05	6.25E+05	1	1.01	9.48E−01	1.16E+00	3.33E+00
$k=50$	4.85E+05	4.84E+05	4.78E+05	1.01	1.01	1.92E+00	2.42E+00	6.30E+00
$k=100$	3.67E+05	3.65E+05	3.59E+05	1.02	1.02	3.07E+00	5.57E+00	6.92E+00

- v3: IKM−+ with $\alpha = 1.0$
- v4: IKM−+ with "If #success > k, then go to end", instead "If #success > $k/2$, then go to end".
- v5: Random initialization for the first use of the k-means, in IKM−+.
- v6: KM++ initialization for the first use of the k-means, in IKM−+.

Results of this experiment are summarized in Tables 4 and 5, which show average SSEDM, and runtime obtained by each version of IKMM−+, in 50 runs. As expected, results of v4 are similar to v1, because it is obvious that all centers cannot be only located in one grand-truth cluster. In one case, for Isolet, v3 is better than v1, however, it is slightly slower than v1Accuracy of IKMM−+ with initialization method in [41] is better than with random or KM++ initialization methods.

4.4. Results on the third type dataset

The third type of datasets includes one dataset, KDDCUP04Bio, which is also a real-world dataset, comes from bioinformatics field, and its number of data points and dimension are large enough, where, in some works [38,39], it is used as a large-scale instance. Its original number of clusters is 2000, which is also rather large. In this experiment, all three competitors are applied to this instance, using variant numbers of clusters including 125, 250, 500, 1000, and 2000. Results of 30 runs of each algorithm for each of mentioned number of clusters are introduced in Table 6. It is obvious that increasing number of clusters decreases the accuracy of KM, in comparison to other competitors, and decreases the speed of KM++ dramatically, where for $k=2000$, KM++ is more than 18 times slower than KM. Fig. 11 shows a better view of runtimes of algorithms. Among these three algorithms, the accuracy of IKM−+ is at least as well as KM++. In addition, IKM−+ has an acceptable runtime.

4.5. More experiments to consider effects of increasing number of clusters

Results of predecessor experiment reveal that for a large-scale dataset like KDDCUP04Bio when the number of clusters is increased, the accuracy of the KM in terms of obtained SSEDM, and speed of KM++ dramatically decrease, while IKM−+ can obtain accurate results in runtime similar to KM. This sub-section reports effects of increasing number of clusters, where competitor algorithms are applied to datasets of the second type. Among these datasets, Iris is too small, so it is ignored in this experiment. Three competitors are applied to these instances, using variant numbers of clusters including 12, 25, 50, and 100. Tables 7–11 show results of 50 runs.

For all cases shown in Tables 7–11, the accuracy of IKM−+, in terms of minimizing SSEDM is higher than KM and KM++. In addition, based on Table 11, results for Statlog, the accuracy of IKM−+ is higher than KM and KM++, with considerable

Table 10
Effects of increasing number of clusters up to 100 on Musk.

	SSEDM			$SSEDM_{IKM-+}$/		Runtime (s)		
	KM	KM++	IKM−+	$SSEDM_{KM}$	$SSEDM_{KM++}$	KM	KM++	IKM−+
$k=12$	3.23E+09	3.23E+09	3.15E+09	1.03	1.03	2.65E−01	3.13E−01	5.17E−01
$k=25$	2.62E+09	2.63E+09	2.53E+09	1.04	1.04	5.75E−01	8.69E−01	1.28E+00
$k=50$	2.13E+09	2.12E+09	2.05E+09	1.04	1.03	1.24E+00	2.37E+00	2.09E+00
$k=100$	1.69E+09	1.69E+09	1.61E+09	1.05	1.05	2.34E+00	7.23E+00	1.83E+00

Table 11
Effects of increasing number of clusters up to 100 on Statlog.

	SSEDM			$SSEDM_{IKM-+}$/		Runtime (s)		
	KM	KM++	IKM−+	$SSEDM_{KM}$	$SSEDM_{KM++}$	KM	KM++	IKM−+
$k=12$	5.60E+08	5.70E+08	2.20E+08	2.54	2.59	7.96E−01	8.82E−01	1.16E+00
$k=25$	3.77E+08	3.30E+08	9.10E+07	4.15	3.62	2.08E+00	1.93E+00	2.73E+00
$k=50$	3.08E+08	1.63E+08	3.00E+07	10.26	5.44	5.05E+00	4.99E+00	8.74E+00
$k=100$	9.98E+07	5.15E+07	1.10E+07	9.11	4.7	1.16E+01	1.36E+01	1.56E+01

Table 12
Effects of increasing number of clusters up to 1000 on Statlog.

	SSEDM			$SSEDM_{IKM-+}$/		Runtime (s)		
	KM	KM++	IKM−+	$SSEDM_{KM}$	$SSEDM_{KM++}$	KM	KM++	IKM−+
$k=125$	9.78E+07	4.14E+07	8.03E+06	12.19	5.16	1.38E+01	1.95E+01	1.14E+01
$k=250$	9.39E+07	1.77E+07	3.78E+06	24.85	4.7	2.68E+01	5.64E+01	1.65E+01
$k=500$	9.25E+07	6.64E+06	2.68E+06	34.48	2.47	4.51E+01	1.83E+02	2.48E+01
$k=1000$	9.05E+07	2.44E+06	1.05E+06	86.35	2.33	6.70E+01	6.80E+02	3.19E+01

differences. According to results of Table 11, for $k=50$, SSEDM obtained by IKM−+, more than 10, and more than 5 times, is smaller than SSEDM of KM and KM++, respectively. observing these results is the main reason to have another experiment on Statlog, using larger numbers of clusters including 125, 250, 500, and 1000. The number of runs in these experiments is 50, and results are introduced on Table 12. Expectedly, IKM−+ can obtain better results with considerable differences. For $k=1000$, SSEDM obtained by IKM−+, more than 86 times, is smaller than SSEDM of KM. It is obvious that for a large number of clusters, KM++ is not satisfactory. In this experiments, for $k=1000$, it is, more than 20 times, slower than IKM−+, and its accuracy is, more than 2 times, lower than IKM−+.

5. Conclusion

The proposed I-k-means−+ tries to iteratively improve the quality of solution produced by the *k*-means via removing one cluster (minus), dividing another one (plus), and applying re-clustering again, in each iteration. Although this process seems to be time-consuming, using some heuristics effectively speeds up the I-k-means−+, where, for the largest data set, with the largest number of clusters in experiments, its runtime is less than 1.22 times slower than the *k*-means, while for many datasets, its accuracy, in terms of obtained *SSEDM*, is considerably higher than both the *k*-means and k-means++. For many datasets, its worst solution is better than the best solution of both the *k*-means and k-means++.

Advantages of I-k-means−+ reveal more when it is applied to instances with a large number of data points or a large number of clusters. For large-scale dataset like KDDCUP04Bio, when the number of clusters is increased, the accuracy of the KM in terms of obtained *SSEDM*, and speed of KM++ dramatically decrease, while IKM−+ can obtain accurate results in runtime similar to KM. For Statlog, a dataset with 58,000 number of data points, for all variants number of clusters from 12 up to 1000, the accuracy of IKM−+ is, at least 2 times, higher than KM++, while for 1000 number of clusters IKM−+ is, more than 21 times, faster than KM++.

In addition to up-coming advantages, perhaps, the proposed iterative minus-plus strategy has the capability to be applied to similar algorithms like FCM with similar objective function, and similar steps to the *k*-means in its algorithm. This case is left as future works, to be considered in details.

References

[1] K. Liao, G. Liu, L. Xiao, C. Liu, A sample-based hierarchical adaptive k-means clustering method for large-scale video retrieval, Knowl.-Based Syst. 49 (2013) 123–133.

[2] J.P. Sarkar, I. Saha, U. Maulik, Rough possibilistic type-2 fuzzy C-means clustering for MR brain image segmentation, Appl. Soft Comput. 46 (2016) 527–536. arXiv:10.1016/j.asoc.2016.01.040.

[3] H. Zhou, Y. Liu, Accurate integration of multi-view range images using k-means clustering, Pattern Recognit. 41 (1) (2008) 152–175.

[4] H.-W. Yoo, S.-H. Jung, D.-S. Jang, Y.-K. Na, Extraction of major object features using VQ clustering for content-based image retrieval, Pattern Recognit. 35 (5) (2002) 1115–1126.

[5] M. Mahdavi, H. Abolhassani, Harmony k-means algorithm for document clustering, Data Min. Knowl. Discov. 18 (3) (2009) 370–391.

[6] A. Chitra, A. Rajkumar, Paraphrase Extraction using fuzzy hierarchical clustering, Appl. Soft Comput. 34 (2015) 426–437.

[7] Y.K. Lam, P.W. Tsang, eXploratory k-means: a new simple and efficient algorithm for gene clustering, Appl. Soft Comput. 12 (3) (2012) 1149–1157.

[8] V. Elyasigomari, M.S. Mirjafari, H.R.C. Screen, M.H. Shaheed, Cancer classification using a novel gene selection approach by means of shuffling based on data clustering with optimization, Appl. Soft Comput. 35 (2015) 43–51.

[9] B. Baranidharan, B. Santhi, DUCF: distributed load balancing unequal clustering in wireless sensor networks using fuzzy approach, Appl. Soft Comput. 40 (2016) 495–506.

[10] S. Qiao, T. Li, H. Li, J. Peng, H. Chen, A new blockmodeling based hierarchical clustering algorithm for web social networks, Eng. Appl. Artif. Intell. 25 (3) (2012) 640–647.

[11] M. Carullo, E. Binaghi, I. Gallo, An online document clustering technique for short web contents, Pattern Recognit. Lett. 30 (10) (2009) 870–876.

[12] X. Cai, W. Li, A spectral analysis approach to document summarization: clustering and ranking sentences simultaneously, Inf. Sci. 181 (18) (2011) 3816–3827.

[13] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, in: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, Portland, Oregon, 1996.

[14] S. Guha, R. Rastogi, K. Shim, CURE: An efficient clustering algorithm for large data sets, in: Proceedings of the ACM SIGMOD Conference, 1998.

[15] G. Karypis, E.-H. (Sam) Han, V. Kumar, Chameleon: hierarchical clustering using dynamic modeling, Computer 32 (8) (1999) 68–75.
[16] É.O. Rodrigues, L. Torok, P. Liatsis, J. Viterbo, A. Conci, k-MS: a novel clustering algorithm based on morphological reconstruction, Pattern Recognit. 66 (2017) 392–403.
[17] J.B. MacQueen, Some methods for classification and analysis of multivariate observations, in: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, California, 1967.
[18] X. Wu, V. Kumar, J. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G.J. McLachlan, A. Ng, B. Liu, P.S. Yu, Z.-H. Zhou, M. Steinbach, D.J. Hand, D. Steinberg, Top 10 algorithms in data mining, Knowl. Inf. Syst. 14 (1) (2008) 1–37.
[19] P. Drineas, A.M. Frieze, R. Kannan, S.S. Vempala, V. Vinay, Clustering large graphs via the singular value decomposition, Mach. Learn. 56 (1–3) (2004) 9–33.
[20] A.K. Jain, Data clustering: 50 years beyond k-means, Pattern Recognit. Lett. 31 (8) (2010) 651–666.
[21] D. Arthur, S. Vassilvitskii, k-means++: the advantages of careful seeding, in: Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, New Orleans, Louisiana, 2007.
[22] M. Erisoglu, N. Calis, S. Sakallioglu, A new algorithm for initial cluster centers in k-means algorithm, Pattern Recognit. Lett. 32 (2011) 1701–1705.
[23] S.J. Redmond, C. Heneghan, A method for initialising the k-means clustering algorithm using kd-trees, Pattern Recognit. Lett. 28 (2007) 965–973.
[24] M.E. Celebi, H.A. Kingravi, A.P. Vela, A comparative study of efficient initialization methods for the k-means clustering algorithm, Expert Syst. Appl. 40 (2013) 200–210.
[25] School of Computing, University of Eastern Finland. (2016) [Online] "https://cs.joensuu.fi/sipu/datasets/.
[26] T.F. Gonzalez, Clustering to minimize the maximum intercluster distance, Theor. Comput. Sci. 38 (2–3) (1985) 293–306.
[27] I. Katsavounidis, C.-C.J. Kuo, Z. Zhang, A new initialization technique for generalized Lloyd iteration, IEEE Signal Process. Lett. 1 (10) (1994) 144–146.
[28] T. Su, J.G. Dy, A deterministic method for initializing k-means clustering, in: Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence, 2004.
[29] A. Likas, N. Vlassis, J.J. Verbeek, The global k-means clustering algorithm, Pattern Recognit. 36 (2) (2003) 451–461.
[30] A.M. Bagirov, Modified global k-means algorithm for minimum sum-of-squares clustering problems, Pattern Recognit. 41 (10) (2008) 3192–3199.
[31] J.Z. Lai, T.-J. Huang, Fast global k-means clustering using cluster membership and inequality, Pattern Recognit. 43 (5) (2010) 1954–1963.
[32] A.M. Bagirov, J. Ugon, D. Webb, Fast modified global k-means algorithm for incremental cluster construction, Pattern Recognit. 44 (4) (2011) 866–876.
[33] G. Tzortzis, A. Likas, The MinMax k-means clustering algorithm, Pattern Recognit. 47 (7) (2014) 2505–2516.
[34] C. Boutsidis, A. Zouzias, M.W. Mahoney, P. Drineas, Randomized dimensionality reduction for k-means clustering, IEEE Trans. Inf. Theory 61 (2) (2015) 1045–1062.
[35] D. Feldman, M. Schmidt, C. Sohler, Turning Big data into tiny data: constant–size coresets for k-means, PCA and projective clustering, in: Proceedings of *the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, 2013.
[36] C. Elkan, Using the triangle inequality to accelerate k-means, in: Proceedings of the Twentieth International Conference on International Conference on Machine Learning, 2003.
[37] G. Hamerly, Making k-means even faster, in: Proceedings of *SIAM International Conference on Data mining (SDM)*, 2010.
[38] X. Jin, S. Kim, J. Han, L. Cao, Z. Yin, GAD: general activity detection for fast clustering on large data, in: Proceedings of *2009 SIAM International Conference on Data Mining*, Sparks, Nevada, USA, 2009.
[39] X. Jin, S. Kim, J. Han, L. Cao, Z. Yin, A general framework for efficient clustering of large datasets based on activity detection, Stat. Anal. Data Min. 4 (2011) 11–29.
[40] J.Z. Lai, Y.-C. Liaw, Improvement of the k-means clustering filtering algorithm, Pattern Recognit. 41 (12) (2008) 3677–3681.
[41] H. Ismkhan, An initialization method for the k-means using the concept of useful nearest centers (2017). arXiv:1705.03613 [cs.LG].

Hassan Ismkhan joined University of Bonab as an Instructor, after he got his Master Degree in computer engineering in 2011. His teaching courses include data structures, algorithm design, information retrieval, and theory of formal languages and automata. His research interests include Social Networks, Information Retrieval, Machine learning, Clustering Algorithms, Evolutionary Algorithms and Operational Research.