DOI: 10.25092/baunfbed.532619 *J. BAUN Inst. Sci. Technol., 21(1), 169-183, (2019)*

An incremental fuzzy algorithm for data clustering problems

Elvin NASİBOV, Burak ORDİN*

Ege University, Faculty of Science, Department of Mathematics, Bornova, İzmir

Geliş Tarihi (Recived Date): 28.03.2018
Kabul Tarihi (Accepted Date): 18.08.2018

Abstract

Data Cluster analysis is an important part of data mining. It can be handled as two types, hard and soft clustering. In hard clustering, a dataset is divided into distinct clusters and each data in the dataset belongs to exactly one cluster. On the contrary data can belong to more than one cluster in soft clustering and each data can be associated with each cluster by a membership degree. Incremental algorithms which are developed for hard clustering have two main advantages. They based on the nonsmooth-nonconvex mathematical model which allows significantly reduce the number of variables and they choose one cluster center for each step that leads to obtain better objective function. In this paper, we propose an incremental fuzzy algorithm for soft clustering problems and present results of numerical experiments on 11 real-world datasets. These results demonstrate that the proposed algorithm is efficient for solving the soft clustering problems.

Keywords: *A fuzzy c-means, global k-means algorithm, fuzzy clustering, nonsmooth optimization.*

Veri kümeleme problemleri için artımlı bir bulanık algoritma

Özet

Veri kümeleme analizi veri madenciliğinin önemli bir parçasıdır. Kesin ve esnek kümeleme olmak üzere iki sınıfta ele alınabilir. Kesin kümelemede bir veri seti kümelere ayrıldığında herbir veriseti içerisindeki her eleman yalnız ve yalnız bir kümeye ait olabilir.Esnek kümelemede ise kesin kümelemenin aksine her bir eleman belirli bir üyelik derecesi ile birden fazla kümeye ait olabilir. Kesin kümeleme için geliştirilmiş olan artımlı algoritmalar iki ana avantaja sahiptir.

Elvin NASİBOV, elvin.nasibov@yahoo.com, https://orcid.org/0000-0002-0105-5447
* Burak ORDİN, burak.ordin@ege.edu.tr, https://orcid.org/0000-0001-7897-3265

Bunlardan birincisi sahip olduğu nonsmooth-nonconvex matematiksel model yapısı sayaseinde değişken sayısı önemli ölçüde azaltılabilir. Bunun yanısıra her bir adımda bir tek küme merkezi seçilerek belirli bir k değerine kadar bu devam ettiği için küresel anlamda amaç fonksiyonu için daha iyi değerler elde edilebilir. Bu çalışmada esnek kümeleme problemlerinin çözümü için artımlı bir bulanık algoritma hazırlanmıştır ve algoritmayla gerçek 11 veriseti üzerinde gerçekleştirilen hesaplama denemelerinin sonuçları sunulmuştur. Sonuçlar esnek kümeleme problemlerinin çözümü için hazırlanan algoritmanın yararlılığını göstermektedir.

Anahtar kelimeler: *Bulanık c-ortalamalar, küresel k-ortalamalar algoritması, nonsmooth optimizasyon.*

1. Introduction

The cluster analysis organizes a collection of patterns into clusters based on similarity. It is also known as the unsupervised classification of patterns and has many applications in different areas such as medicine, business, engineering systems, image processing and etc.

Clustering problem can be examined in various viewpoints. Accordingly, one of them is to divide clustering problem as hard and soft clustering. Each point of the data set belongs to exactly one cluster that is known as hard clustering. If data elements can belong to more than one cluster it is called as soft clustering. In soft clustering, a data element is connected to each cluster with a membership degree. This idea firstly suggested by Zadeh in fuzzy set theory [1].

The clustering problem can be considered as a global optimization problem. So, it may have many solutions and only global solution provides the best cluster structure of a dataset. To solve this problem, a great deal of optimization techniques have been developed for last twenty years. These techniques are approaches from differerent algorithmic methods as interior point methods, branch and bound, cutting plane, the variable neighborhood search algorithm, dynamic programming, and metaheuristics like tabu search, genetic algorithms, simulated annealing, [2-14]. Especially, incremental methods have some advantages as nonsmooth and nonconvex mathematical model that has less variables and an approach which leads to a global or near global minimizer in hard clustering.

These algorithms attempt to optimally add one new cluster center at each stage. To compute k-partition of a dataset, these algorithms start from an initial state with the $(k-1)$ centers for the $(k-1)$ clustering problem and try to find best k-th center.

The global k-means (GKM) algorithm, introduced in Ref. [15], is a notable developed version of the k-means algorithm. In the GKM algorithm, which is an incremental algorithm, each data point is used as a starting point for the k-th cluster center. The main purpose of GKM algorithm leads at least to a near global minimizer.

The paper [16] introduces the modified global k-means algorithm. It is also an incremental algorithm and subproblems are solved iteratively in the modified GKM algorithm to compute initial solutions for cluster centers. Computational experiments

presented in [16] demonstrate that the modified global *k*-means algorithm has ability to find a better solutions than the global *k*-means algorithm.

In 2010 Vanisri D. and Loganathan developed new algorithm to compute initial cluster centers for *k*-means clustering. According to this method, data is partitioned by a cutting plane algorithm in a cell that divides cell in to two smaller cells. Cells are partitioned one at a time till the number of cells equals to the predefined number of clusters *k*. The centers of the *k* cells become the initial cluster centers for *k*-means. The computational experiments show that the proposed algorithm is effective, converge to better clustering results than those of the random initialization method [17].

Besides many algorithms have been developed as different versions of *k*-means and global *k*-means ([18], [19]).

There is very often no sharp boundary between clusters so that fuzzy clustering is more appropirate in real applications. So, many researches have studied on fuzzy clustering.

In one of these studies presents a fuzzy c-means (FCM) algorithm that deals with spatial information into the membership function for clustering [20]. The expressed function in this method is the summation of the membership function in the neighborhood of each pixel under the consideration.

To evaluate volumes of cerebrospinal fluid (CSF), white matter, and gray matter from transaxial magnetic resonance images (MRI) of the brain are described a new algorithm in [21]. The algorithm was able to detect the expected increased amounts of CSF and decreased amounts of white matter characteristic of the hydrocephalic brain.
Antonino Staiano and others describe a novel approach to fuzzy clustering, which organizes the data in clusters on the basis of the input data and a ‘prototype’ regression function built, in the output space, as a summation of a number of linear local regression models. They especially claim the given algorithm to be effective in the training of RBFNNs leading to improved performance with respect to other clustering algorithms [22]. Here supervised fuzzy clustering algorithm is using to search for the centroids of the hidden units of the RBFNNs.

In 2010 Velmurugan and Santhanam working on implementation and performance of two clustering algorithms namely centroid based k-means and object based c-means. They show that the behavior of both algorithms depends on the number of data points as well as on the number of clusters. The input data points are generated by using normal distribution and another by applying uniform. The running time for each algorithm is analyzed and the computational experiments are compared with each other [23].

The purpose of this study is to obtain an incremental algorithm in fuzzy clustering problem which provides better solutions. So, we propose an incremantal fuzzy clustering algorithm that based on global *k*-means. The results of numerical experiments on 12 datasets demonstrate the superiority of the proposed algorithm over the fuzzy clustering algorithm.

The rest part of the paper is organized as follows: Section 2 gives mathematical models for clustering problem. Two algorithms for clustering problem are presented in Section 3. A Modified Global k-Means Algorithm on a Fuzzy Model is expressed in section 4.

The results of numerical experiments are given in Section 5. Section 6 concludes the paper.

2. Mathematical models for clustering problem

This section adresses various mathematical models of clustering problem. Let us take a finite set of points A in the n-dimensional space R^n, that is $A = \{a^1, \dots, a^m\}$, where $a^i \in R^n$, $i = 1, \dots, m$.

The hard uncontrained partition clustering problem is the distribution of the points of the set A into a given number k of disjoint subsets A^j ; j = 1,...,k with respect to predefined criterias:

- $A^j \neq \emptyset$, $j = 1, \dots, k$;
- $A^j \cap A^l = \emptyset$, $j, l = 1, \dots, k$, $j \neq l$;
- $A = \bigcup_{j=1}^{k} A^j$;
- No constrains are imposed on the clusters A^j, $j = 1, \dots, k$.

The sets A^j , j=1,...,k are called clusters. Data points from the same cluster are similar and data points from different clusters are dissimilar to each other. Similarity of data points is defined by the so-called similarity measure. This measure can be defined by a distance between data points and their cluster centers. Let $d(x ; y)$ be a distance between points x and y. Then the clustering problem can be reduced to the following optimization problem (see [24]):

$$\text{Min} \quad \psi_k(x, w) = \frac{1}{m} \sum_{i=1}^{m} \sum_{j=1}^{k} w_{ij} d\left(x^j, a^i\right) \tag{1}$$

$$\text{subject to } x = (x^1, \dots, x^k) \in R^{n \times k}, \tag{2}$$

$$\sum_{j=1}^{k} w_{ij} = 1 \,, i = 1, \dots, m, \tag{3}$$

$$w_{ij} = 0 \text{ or } 1, \quad i = 1, \dots, m, \quad j = 1, \dots, k \tag{4}$$

where w_{ij} is the association weight of pattern a^i with the cluster j, given by

$$w_{ij} = \begin{cases} 1, & if\ pattern\ a^i\ is\ allocated\ to\ the\ cluster\ j \\ 0, & otherwise \end{cases} \tag{5}$$

$$\text{and} \quad x^j = \frac{\sum_{i=1}^{m} w_{ij} a^i}{\sum_{i=1}^{m} w_{ij}}, \ j = 1, \dots, k. \tag{6}$$

Here w is a $m \times k$ matrix.

The distance d can be defined using different norms. In this paper we use the following squared euclidean norm.

$$d(x, y) = \left(\sum_{i=1}^{n} |x_i - y_i|^2 \right)^{1/2}$$

The problem (1)-(6) is called the mixed integer nonlinear programming formulation of the clustering problem.
Nonsmooth nonconvex optimization formulation of the clustering problem (1)-(6) is as follows (see [24]):

$$min\ f_k(x)\ \ subject\ to\ (x^1, \dots, x^k)\ \epsilon\ R^{n\times k}, \tag{7}$$

$$where$$

$$f_k(x^1, \dots, x^k) = \frac{1}{m}\sum_{i=1}^{m} min_{j=1,\dots,k}\ d\left(x^j, a^i\right) \tag{8}$$

ψ_k and f_k functions are called cluster functions. The objective function ψ_k depends on both binary (w_{ij}, i=1,...,m, j=1,...,k) and continuous variables (x_1,...,x_k), whereas the function f_k depends only on continuous variables x_1,..., x_k. According to this both functions are nonconvex. The number of variables in problem (1) - (4) is $(m + n) \times k$ and in problem (6) this number is only $n \times k$ and the number of variables does not depend on the number of instances. Problems (1) - (4) and (6) are equivalent in the sense that their global minimizers coincide.

In the soft unconstrained partition clustering problem, the distribution of the points of the set A into k clusters A^j , j=1,...,k carry out with memberships. So data points have a membership degree for each cluster. That is, a particular point can belong to more than one cluster with certain possibility. Fuzzy clustering is often best suited as there is often no sharp boundary between clusters for the data. In fuzzy clustering, membership degrees between zero and one are used instead of crisp assignments of the data to clusters in real applications. This method was developed by Dunn in 1973 and improved by Bezdek in 1981 and it is frequently used in pattern recognition [25].

It is based on minimization of the following objective function:

$$\psi_k = \sum_{i=1}^{m}\sum_{j=1}^{k} u_{ij}^q \left\|a^i - x^j\right\|^2, 1 < q < \infty$$

where q is any real number greater than 1, u_{ij} is the degree of membership of a d-dimensional measured data a^i in the cluster j, x^j is the d-dimensional center of the cluster j, and ||*|| is any norm expressing the similarity between any measured data and the cluster center. Fuzzy partitioning is carried out through an iterative optimization of the objective function shown above, with the update of membership u_{ij} and the cluster centers x_j by:

$$u_{ij} = \frac{1}{\sum_{l=1}^{k}\left(\frac{\left\|a^i - x^j\right\|}{\left\|a^i - x^l\right\|}\right)^{\frac{2}{q-1}}}, \quad \text{and} \quad x^j = \frac{\sum_{i=1}^{m} u_{ij}^q . a^i}{\sum_{i=1}^{m} u_{ij}^q}.$$

This iteration will stop when $\left\|U^{(t+1)} - U^{(t)}\right\| < ʔ$, where $ʔ$ is a termination criterion near to 0, whereas t is the iteration steps. This procedure converges to a local minimum or a saddle point of ψ_k.

The following section handles an algorithm based on incremental and for fuzzy clustering problem in a nonsmooth nonconvex mathematical model.

3. Algorithms for hard and soft clustering problems

In this section we give the global *k*-means algorithm which is a basic incremantal algorithm in hard clustering. Also we express fuzzy *k*-means algorithm in soft clustering.

The global *k*-means algorithm proceeds as follows.

Algorithm 1. The global k-means algorithm.
Step 1. (Initialization) Compute the centroid x^1 of the set A:

$$x^1 = \frac{1}{m}\sum_{i=1}^{m} a^i,\ a^i \in A,\ i = 1, \dots, m$$

and set t=1.
Step 2. (Stopping criterion) Set $t = t + 1$. If $t > k$, then Stop.
Step 3. Take the centers $x^1, x^2, \dots, x^{t-1}$ from the previous iteration and consider each point a^i of A as a starting point for the t-th cluster center, thus obtaining m initial solutions with t points $(x^1, \dots, x^{t-1}, a)$; apply the *k*-means algorithm to each of them; keep the best t-partition obtained and it's centers $y^1, y^2, \dots, y^t$.
Step 4. Set $x^i = y^i$, i=1,...,t and goto step 2.

The modified global *k*-means algorithm, introduced in [8], is also an incremental clustering algorithm. The algorithm starts with the computation of one cluster center, that is with the computation of the centroid of the dataset, and attempts to optimally add one new cluster center at each iteration for solving *k*-partition problem,. As using the (*k*-1) centers for the (*k*-1)-partition problem and the remaining *k*-th center is placed in an appropriate place to solve k-partition problem. An auxiliary cluster function is defined using *k*-1 cluster centers from the (*k*-1)-th iteration and is minimized to compute the starting point for the *k*-th center. Then this new center together with previous *k*-1 cluster centers is an input as a starting point for the *k*-partition problem. This is a good idea for to find a global or a near global solution to hard clustering problem.

The general scheme of fuzzy k-means algorithms for finding the k-partition of the set A is as follows:

Algorithm 2. The fuzzy *k*-means algorithm
Step 1. Initialize randomly $U = [u_{ij}]$ matrix denoted as $U^{(0)}$. Set iteration number: $t = 0$.
Step 2. For the iteration t, calculate the centers x^j, $j = 1..k$, according to the precomputed $U^{(t)}$.
Step 3. Update membership degrees according to recently computed cluster centers, and denote as $U^{(t+1)}$.
Step 4. If $\left\|U^{(t+1)} - U^{(t)}\right\| < ?$ then Stop; otherwise set $t = t + 1$, and return to step 2.

In fuzzy *k*-means algorithm, the centroid of a cluster is computed as being the mean of all points, weighted by their degree of belonging to the cluster and the degree of being in a certain cluster is related to the inverse of the squared distance to the cluster.

By using the above mentioned algorithms, we obtain an incremental fuzzy algorithm for soft clustering problem in the next section.

4. An incremental fuzzy algorithm for clustering problems

In this section we describe an incremental fuzzy algorithm for solving soft clustering problems. Our algorithm builds clusters dynamically adding one cluster center at a time. The incremental fuzzy clustering for finding the *k*-partition of the set A proceeds as follows:

Algorithm 3. A fuzzy global *k*-means algorithm
Step 1. Initialize $U = [u_{ij}]$ matrix, $U^{(0)}$. Compute the centroid x^1 of the set A:

$$x^1 = \frac{1}{m}\sum_{i=1}^{m} a^i,\ a^i \in A,\ i = 1, \dots, m$$

and set t=2.
Step 2. Take the centers $x^1, x^2, \dots, x^{t-1}$ from the previous iteration and consider each point a of A as a starting point for the t-th cluster center, thus obtaining m initial solutions with t points $(x^1, \dots, x^{t-1}, a)$; apply the fuzzy *k*-means (with $k \equiv t$) algorithm to each of them; keep the best t-partition obtained and its centers $y^1, y^2, \dots, y^t$.
Step 3. Set $x^i = y^i$, i=1,...,t; Set $t = t + 1$; if $t \leq k$, go to step 2 else Stop.

The aim of this algorithm is to gather advantages of incremental algorithms in hard clustering for fuzzy clustering. The algorithm starts with the computation of one cluster center, that is with the computation of the centroid of the dataset, and attempts to optimally add one new cluster center at each iteration for solving a subproblem. Each new center together with previous *k*-1 cluster centers is an input as a starting point for the subproblem.

5. Computational experiments

To verify the efficiency of the proposed algorithm numerical experiments with a number of real-world datasets have been carried out on a PC Intel(R) Core(TM) i5 with CPU 2.90 GHz and RAM 6 GB. We also compare the proposed algorithm with the fuzzy *k*-means algorithm. 11 datasets have been used in numerical experiments [26]. The brief description of the datasets is given in Table 1.

Table 1. The brief description of datasets.

Datasets	Number of instances	Number of attributes
BAVARIA1	89	3
BAVARIA2	89	4
IRIS	150	4
WINE	178	14
CLEVELAND	297	13

Table 1. (Continued).

LIVER	345	6
IONOSPHERE	351	34
DIABETS	768	8
TSP	1060	2
PAGE BLOCK	5473	10
PENDIGIT	10992	16

The following notations are used in the numerical experiments.

- k is the number of clusters;
- f_{opt} is the best known value of the cluster ;
- f is average value of the clustering for FCM algorithm after calculating 10 times and the only value for FGMCM (results are the same for every time calculating by FGMCM – this is global solution)
- E is the error in %;
- t is the CPU time;
- D_{ave} is average value of calculated distances for FCM after calculating 10 times and the only value for FGMCM (results are the same for every time calculating by FGMCM – this is global solution)
- FGMCM stands for the proposed algorithm.

The error E is computed as

$$E = \frac{(\bar{f} - f_{opt})}{f_{opt}} \cdot 100,$$

where $\bar{f}$ is the best value of the objective function obtained by an algorithm.
Results for small datasets, bavaria 1, bavaria 2, irish plants, are presented in tables 2, 3, 4.

Table 2. Results for dataset Bavaria1.

k	**FCM (average of 10 times running)**				**FGMCM**			
	f	E	t	D_{ave}	f	E	t	D_{ave}
2	$99.3 * 10^{10}$	71.63	00:00:00.5500000	445	$579.00 * 10^{9}$	0.00	00:00:00.5530413	356
3	$39 * 10^{10}$	71.06	00:00:00.5600000	3123.9	$228.00 * 10^{9}$	0.00	00:00:00.9480302	7031
4	$15 * 10^{10}$	25.79	00:00:00.5700000	7796.4	$119.00 * 10^{9}$	0.00	00:00:00.8381701	13439
5	$12.8 * 10^{10}$	174.01	00:00:00.5270000	7431.5	$46.80 * 10^{9}$	0.00	00:00:01.1524557	21894
6	$11.5 * 10^{10}$	269.90	00:00:00.6100000	12762.6	$31.00 * 10^{9}$	0.00	00:00:01.4022011	32040
7	$11.4 * 10^{10}$	379.07	00:00:00.6400000	18752.3	$23.80 * 10^{9}$	0.00	00:00:01.9470050	40762
8	$6.84 * 10^{10}$	227.18	00:00:00.6270000	22570.4	$20.90 * 10^{9}$	0.00	00:00:02.1952856	45746
9	$4.86 * 10^{10}$	161.19	00:00:00.6500000	25712.1	$18.60 * 10^{9}$	0.00	00:00:02.4449404	52154
10	$10.1 * 10^{10}$	511.86	00:00:00.5800000	14952	$16.50 * 10^{9}$	0.00	00:00:02.6445966	67284
11	$7.65 * 10^{10}$	451.52	00:00:00.6570000	27901.5	$13.90 * 10^{9}$	0.00	00:00:03.1785964	73158
12	$8.23 * 10^{10}$	550.79	00:00:00.6800000	40584	$12.60 * 10^{9}$	0.00	00:00:03.5773653	91314
13	$7.35 * 10^{10}$	516.36	00:00:00.6500000	32743.1	$11.90 * 10^{9}$	0.00	00:00:03.9459486	100570
14	$8.36 * 10^{10}$	679.65	00:00:00.7530000	30651.6	$10.70 * 10^{9}$	0.00	00:00:04.3622616	122998
15	$3.56 * 10^{10}$	274.26	00:00:00.7370000	82770	$9.51 * 10^{9}$	0.00	00:00:04.6280423	209773
20	$2.27 * 10^{10}$	310.65	00:00:00.7870000	75116	$5.52 * 10^{9}$	0.00	00:00:08.8072048	511750
25	$2.18 * 10^{10}$	408.66	00:00:00.9370000	138840	$4.28 * 10^{9}$	0.00	00:00:11.2148460	800021
30	$1.52 * 10^{10}$	7503.97	00:00:00.9600000	134301	$0.20 * 10^{9}$	0.00	00:00:19.7241270	1624517
35	$0.475 * 10^{10}$	3767.35	00:00:01.2130000	223034	$0.12 * 10^{9}$	0.00	00:00:25.4283391	2744671
40	$0.908 * 10^{10}$	8417.33	00:00:01.2770000	262728	$0.11 * 10^{9}$	0.00	00:00:30.0118952	3716462
45	$0.52 * 10^{10}$	7081.97	00:00:01.4170000	289161	$0.07 * 10^{9}$	0.00	00:00:41.7922911	4356906
50	$0.579 * 10^{10}$	9446.19	00:00:01.5030000	344875	$0.06 * 10^{9}$	0.00	00:00:38.7687368	5464689

Table 3. Results for dataset Bavaria2.

k	FCM (average of 10 times running)				FGMCM			
	F	E	t	D_{ave}	f	E	t	D_{ave}
2	$82.36 * 10^9$	99.72	00:00:00.5600000	623	$41.24 * 10^9$	0.00	00:00:00.6280775	534
3	$20.36 * 10^9$	9.92	00:00:00.6130000	5713.8	$18.52 * 10^9$	0.00	00:00:00.3438855	3471
4	$19.26 * 10^9$	214.95	00:00:00.5930000	5233.2	$6.11 * 10^9$	0.00	00:00:00.5036629	20915
5	$9.17 * 10^9$	137.06	00:00:00.6130000	10724.5	$3.87 * 10^9$	0.00	00:00:00.7406861	36045
6	$8.09 * 10^9$	190.63	00:00:00.6630000	8917.8	$2.78 * 10^9$	0.00	00:00:00.9049078	45123
7	$9.29 * 10^9$	321.79	00:00:00.6600000	12709.2	$2.20 * 10^9$	0.00	00:00:01.2423689	86241
8	$1.79 * 10^9$	0.00	00:00:00.6630000	30117.6	$1.97 * 10^9$	10.05	00:00:01.5564978	88377
9	$1.71 * 10^9$	1.34	00:00:00.6500000	23949.9	$1.69 * 10^9$	0.00	00:00:01.7100315	99591
10	$7.03 * 10^9$	388.00	00:00:00.6600000	21271	$1.44 * 10^9$	0.00	00:00:02.2136175	147651
11	$4.69 * 10^9$	247.89	00:00:00.9100000	28978.4	$1.35 * 10^9$	0.00	00:00:02.4434161	151567
12	$7.75 * 10^9$	564.92	00:00:00.7170000	20185.2	$1.17 * 10^9$	0.00	00:00:02.9141849	188947
13	$3.38 * 10^9$	238.55	00:00:00.6800000	33784.4	$1.00 * 10^9$	0.00	00:00:03.6026968	245640
14	$3.36 * 10^9$	256.84	00:00:00.6370000	31773	$0.94 * 10^9$	0.00	00:00:03.7007947	270560
15	$3.25 * 10^9$	270.96	00:00:00.6830000	46324.5	$0.88 * 10^9$	0.00	00:00:04.1765584	297260
20	$3.04 * 10^9$	350.57	00:00:00.7200000	55002	$0.67 * 10^9$	0.00	00:00:07.0666346	520650
25	$0.81 * 10^9$	47.23	00:00:00.9970000	150187.5	$0.55 * 10^9$	0.00	00:00:11.6787706	1288097
30	$1.08 * 10^9$	125.60	00:00:01.0700000	180225	$0.48 * 10^9$	0.00	00:00:16.1523527	1706664
35	$0.55 * 10^9$	31.55	00:00:01.0900000	185031	$0.42 * 10^9$	0.00	00:00:21.3133292	2628882
40	$0.85 * 10^9$	126.34	00:00:01.4000000	196156	$0.38 * 10^9$	0.00	00:00:28.3237560	3768883
45	$0.42 * 10^9$	28.66	00:00:01.3300000	251514	$0.32 * 10^9$	0.00	00:00:35.6682668	4565967
50	$0.13 * 10^9$	0.00	00:00:01.4030000	266555	$0.29 * 10^9$	116.46	00:00:42.4022704	5305913

Table 4. Results for dataset IRIS.

k	FCM (average of 10 times running)				FGMCM			
	f	E	t	D_{ave}	f	E	t	D_{ave}
2	340.16	163.90	00:00:00.2270000	780	128.90	0.00	00:00:00.9375454	1500
3	151.82	150.91	00:00:00.2200000	3600	60.51	0.00	00:00:01.5426924	35250
4	92.31	121.56	00:00:00.2900000	15900	45.82	0.00	00:00:02.0846915	37050
5	44.20	30.86	00:00:00.3100000	29850	38.72	0.00	00:00:02.0913297	38550
6	43.88	77.45	00:00:00.2800000	33480	33.22	0.00	00:00:02.6636480	40350
7	37.31	83.01	00:00:00.2770000	24150	20.59	0.00	00:00:03.1572483	71850
8	19.84	13.02	00:00:00.3270000	33120	17.67	0.00	00:00:03.7352598	82650
9	22.75	47.68	00:00:00.3400000	45360	15.40	0.00	00:00:04.3483871	165000
10	19.10	36.84	00:00:00.3900000	79500	13.67	0.00	00:00:04.8610650	223500
11	24.67	104.71	00:00:00.3400000	41415	12.05	0.00	00:00:06.0355337	358800
12	16.54	48.41	00:00:00.4070000	84420	11.73	0.00	00:00:06.0276412	364200
13	19.00	83.64	00:00:00.4600000	99840	10.07	0.00	00:00:07.5465784	473400
14	9.44	2.07	00:00:00.6200000	183120	9.23	0.00	00:00:07.9796838	588900
15	15.52	77.23	00:00:00.5000000	106875	8.69	0.00	00:00:08.7852087	703650
20	6.28	0.00	00:00:00.6200000	170100	6.57	4.76	00:00:16.1532486	1217550
25	5.39	1.70	00:00:00.8400000	294000	5.36	0.00	00:00:21.1573325	1631550
30	4.41	2.91	00:00:00.8070000	243450	4.28	0.00	00:00:28.9922859	3335700
35	3.24	0.00	00:00:01.2930000	474075	3.50	10.43	00:00:37.5927287	4514550
40	4.13	33.49	00:00:01.0370000	324600	3.07	0.00	00:00:45.6548817	5996400
45	2.42	0.00	00:00:01.6930000	675675	2.79	12.33	00:00:55.5718831	8160300
50	2.06	0.00	00:00:01.6630000	655500	2.51	19.26	00:01:12.1013739	10182000

These results clearly demonstrate that for small datasets the proposed algorithm finds either better or very similar solutions in comparison with average FCM algorithm. This algorithm fails a few time for Bavaria2 and IRIS, but generally gives much more good solutions than average FCM on small datasets. CPU compution time for both algorithm is not very different. According to that this algorithm can be considered as an alternative for FCM algorithm for small datasets.

Results for medium size datasets are presented in Tables 5- 9.

Table 5. Results for dataset WINE.

k	FCM (average of 10 times running)				FGMCM			
	f	E	t	D_{ave}	f	E	t	D_{ave}
2	$558.74 * 10^4$	50.60	00:00:00.9070000	783.2	$371.01 * 10^4$	0.00	00:00:01.6639851	1780
3	$177.64 * 10^4$	0.31	00:00:00.9430000	17355	$177.09 * 10^4$	0.00	00:00:01.7457751	42898
4	$107.68 * 10^4$	14.35	00:00:01.0570000	38519.2	$94.17 * 10^4$	0.00	00:00:02.7418371	55714
5	$65.26 * 10^4$	0.35	00:00:01.2030000	66216	$65.03 * 10^4$	0.00	00:00:04.0991923	114454
6	$45.87 * 10^4$	0.85	00:00:01.0930000	50196	$45.49 * 10^4$	0.00	00:00:05.6628581	139018
7	$41.59 * 10^4$	4.49	00:00:01.2170000	67159.4	$39.80 * 10^4$	0.00	00:00:07.3178803	151478
8	$33.18 * 10^4$	0.00	00:00:01.2670000	72481.6	$33.92 * 10^4$	2.23	00:00:09.4445816	195622
9	$29.35 * 10^4$	0.00	00:00:01.3830000	97401.6	$30.78 * 10^4$	4.86	00:00:12.3898409	397474
10	$20.58 * 10^4$	44.39	00:00:01.4770000	126380	$14.25 * 10^4$	0.00	00:00:15.2490709	598614
11	$16.79 * 10^4$	31.20	00:00:01.4230000	117088.4	$12.80 * 10^4$	0.00	00:00:18.8798232	831616
12	$17.88 * 10^4$	50.11	00:00:01.8670000	233678.4	$11.91 * 10^4$	0.00	00:00:21.7609895	902104
13	$9.57 * 10^4$	10.19	00:00:01.8970000	231168.6	$8.68 * 10^4$	0.00	00:00:26.0773774	1258460
14	$13.15 * 10^4$	56.76	00:00:01.4130000	100427.6	$8.39 * 10^4$	0.00	00:00:29.9118022	1340696
15	$8.00 * 10^4$	8.38	00:00:01.9730000	247776	$7.38 * 10^4$	0.00	00:00:33.3743522	1434146
20	$5.10 * 10^4$	0.00	00:00:02.2270000	302244	$5.18 * 10^4$	1.71	00:00:57.4538352	2306168
25	$3.89 * 10^4$	0.00	00:00:02.0100000	223390	$4.82 * 10^4$	24.07	00:01:30.7781916	4970650
30	$2.30 * 10^4$	0.00	00:00:03.5400000	525456	$4.61 * 10^4$	100.40	00:02:06.0206362	6861544
35	$1.88 * 10^4$	0.00	00:00:04.0470000	765667	$4.09 * 10^4$	117.17	00:02:47.5433922	8542576
40	$1.52 * 10^4$	0.00	00:00:04.6930000	947672	$3.37 * 10^4$	121.49	00:03:47.4724369	14463568
45	$1.28 * 10^4$	0.00	00:00:05.5900000	1198296	$3.00 * 10^4$	135.23	00:04:49.4770356	17905554
50	$1.07 * 10^4$	0.00	00:00:05.5200000	1172130	$2.51 * 10^4$	135.26	00:05:46.8838754	20414820

Table 6. Results for dataset CLEVELAND.

k	FCM (average of 10 times running)				FGMCM			
	f	E	t	D_{ave}	f	E	t	D_{ave}
2	319.34	$2.23 * 10^2$	00:00:00.8430000	1485	$98.77 * 10^0$	0.00	00:00:03.3924531	1188
3	118.93	$322.00 * 10^2$	00:00:01.1970000	3118.5	$0.37 * 10^0$	0.00	00:00:03.3290343	2970
4	34.29	$138.00 * 10^{27}$	00:00:01.7030000	7484.4	$0.25 * 10^{-25}$	0.00	00:00:05.4786372	6534
5	4.86	$19.60 * 10^{27}$	00:00:01.4570000	12028.5	$0.25 * 10^{-25}$	0.00	00:00:08.0853441	9504
6	21.74	$134.00 * 10^{27}$	00:00:01.6100000	15147	$0.16 * 10^{-25}$	0.00	00:00:11.5452714	13068
7	1.81	$12.10 * 10^{27}$	00:00:01.6000000	16839.9	$0.15 * 10^{-25}$	0.00	00:00:14.7463253	17226
8	14.73	$148.00 * 10^{27}$	00:00:01.6530000	14256	$0.10 * 10^{-25}$	0.00	00:00:20.2609879	24354
9	1.26	$15.30 * 10^{27}$	00:00:01.5870000	19780.2	$0.08 * 10^{-25}$	0.00	00:00:23.7485486	29700
10	1.47	$19.30 * 10^{27}$	00:00:01.6530000	21087	$0.08 * 10^{-25}$	0.00	00:00:30.4869915	44550
11	1.64	$21.50 * 10^{27}$	00:00:01.6630000	30709.8	$0.08 * 10^{-25}$	0.00	00:00:35.1628026	51084
12	1.43	$40.30 * 10^{27}$	00:00:01.6630000	32432.4	$0.04 * 10^{-25}$	0.00	00:00:42.1137532	76032
13	1.04	$29.10 * 10^{27}$	00:00:01.7430000	33976.8	$0.04 * 10^{-25}$	0.00	00:00:49.6678915	83754
14	0.99	$19.20 * 10^{27}$	00:00:01.9170000	35758.8	$0.05 * 10^{-25}$	0.00	00:00:56.1926109	117018
15	0.29	$5.71 * 10^{27}$	00:00:01.6730000	28066.5	$0.05 * 10^{-25}$	0.00	00:01:02.1736826	125928
20	0.23	$10.10 * 10^{27}$	00:00:01.9230000	37422	$0.02 * 10^{-25}$	0.00	00:01:47.2848947	225720
25	0.30	$9.63 * 10^{27}$	00:00:02.1030000	53460	$0.03 * 10^{-25}$	0.00	00:02:58.2028488	3590136
30	0.09	$4.16 * 10^{27}$	00:00:02.2570000	56133	$0.02 * 10^{-25}$	0.00	00:04:11.9694065	7576470
35	0.00	0.00	00:00:02.0730000	60291	$0.02 * 10^{-25}$	0.00	00:05:42.8086813	12821787
40	0.26	$13.80 * 10^{27}$	00:00:02.2270000	91476	$0.02 * 10^{-25}$	0.00	00:07:28.2022964	18659322
45	0.00	0.00	00:00:02.2770000	77517	$0.02 * 10^{-25}$	0.00	00:09:10.5472465	19073637
50	0.00	0.00	00:00:02.3600000	86130	$0.02 * 10^{-25}$	8.14	00:11:01.4780363	19442511

Table 7. Results for dataset LIVER.

k	FCM (average of 10 times running)				FGMCM			
	f	E	t	D_{ave}	F	E	t	D_{ave}
2	$36.11 * 10^4$	2.51	00:00:02.0700000	1932	$35.22 * 10^4$	0.00	00:00:02.3444286	2760
3	$20.87 * 10^4$	0.40	00:00:01.9700000	45643.5	$20.79 * 10^4$	0.00	00:00:03.3543936	74175
4	$15.16 * 10^4$	2.77	00:00:02.3230000	49818	$14.75 * 10^4$	0.00	00:00:03.6011962	82455
5	$11.47 * 10^4$	1.66	00:00:02.0530000	61237.5	$11.29 * 10^4$	0.00	00:00:06.3070420	229080
6	$9.23 * 10^4$	0.00	00:00:02.1600000	123372	$9.23 * 10^4$	0.02	00:00:06.7393495	311880
7	$7.96 * 10^4$	1.37	00:00:02.1900000	80419.5	$7.86 * 10^4$	0.00	00:00:10.2360135	343275
8	$6.83 * 10^4$	0.50	00:00:02.4100000	160632	$6.79 * 10^4$	0.00	00:00:18.2921962	456435
9	$6.04 * 10^4$	0.75	00:00:02.3270000	156802.5	$5.99 * 10^4$	0.00	00:00:14.3302942	807300
10	$5.49 * 10^4$	1.91	00:00:02.5470000	211140	$5.39 * 10^4$	0.00	00:00:16.3938192	1052250
11	$4.87 * 10^4$	0.00	00:00:02.5430000	229977	$4.89 * 10^4$	0.53	00:00:21.6681558	1192665
12	$4.49 * 10^4$	2.61	00:00:02.9200000	421038	$4.37 * 10^4$	0.00	00:00:24.6618096	1975125
13	$4.13 * 10^4$	2.76	00:00:02.6830000	275827.5	$4.02 * 10^4$	0.00	00:00:27.6341652	2441565
14	$3.84 * 10^4$	2.58	00:00:02.9530000	391230	$3.74 * 10^4$	0.00	00:00:30.1996937	2552655
15	$3.72 * 10^4$	6.67	00:00:02.5300000	186817.5	$3.49 * 10^4$	0.00	00:00:34.0220685	2925255
20	$2.69 * 10^4$	4.57	00:00:04.2100000	917010	$2.57 * 10^4$	0.00	00:01:00.6380743	5621085
25	$2.10 * 10^4$	2.74	00:00:04.7100000	1113487.5	$2.04 * 10^4$	0.00	00:01:32.7546106	10701210
30	$1.75 * 10^4$	3.24	00:00:06.1630000	1687050	$1.70 * 10^4$	0.00	00:02:03.4281892	14406510
35	$1.51 * 10^4$	3.36	00:00:07.2170000	2169877.5	$1.46 * 10^4$	0.00	00:02:42.0318315	20134200
40	$1.32 * 10^4$	2.08	00:00:06.9530000	2042400	$1.29 * 10^4$	0.00	00:03:27.3590131	24714420
45	$1.15 * 10^4$	1.14	00:00:07.7770000	2376877.5	$1.13 * 10^4$	0.00	00:04:41.2715947	39758145
50	$1.04 * 10^4$	2.90	00:00:07.9070000	2178675	$1.01 * 10^4$	0.00	00:06:21.4615916	63304050

Table 8. Results for dataset IONOSPHERE.

k	FCM (average of 10 times running)				FGMCM			
	f	E	t	D_{ave}	f	E	t	D_{ave}
2	31.75	14.72	00:00:01.7370000	1404	27.68	0.00	00:00:04.5004389	1404
3	16.95	107.82	00:00:01.8870000	2106	8.16	0.00	00:00:09.7954798	35100
4	10.70	117.54	00:00:02.1770000	2808	4.94	0.00	00:00:17.8372711	183924
5	9.05	249.33	00:00:02.0400000	3510	2.59	0.00	00:00:27.7795093	385749
6	5.42	196.28	00:00:01.8170000	4212	1.83	0.00	00:00:38.8232925	453141
7	3.91	220.60	00:00:01.8300000	4914	1.22	0.00	00:00:51.5455842	541593
8	3.96	344.87	00:00:01.9570000	5616	0.89	0.00	00:01:07.1292686	701649
9	2.88	332.88	00:00:01.9370000	6318	0.66	0.00	00:01:26.8850087	1371357
10	2.17	316.95	00:00:01.9930000	7020	0.52	0.00	00:01:44.8668367	1522287
11	2.24	480.95	00:00:02.0530000	7722	0.39	0.00	00:02:08.2191856	2004912
12	1.70	403.31	00:00:02.0700000	8424	0.34	0.00	00:02:35.1415130	2902068
13	1.87	587.76	00:00:01.9970000	9126	0.27	0.00	00:03:03.7613350	3969810
14	0.88	273.56	00:00:02.0070000	9828	0.23	0.00	00:03:35.9019093	5060718
15	1.47	580.17	00:00:02.5100000	10530	0.22	0.00	00:04:02.2875618	5108103
20	0.73	546.47	00:00:02.2870000	14040	0.11	0.00	00:07:05.4621985	9009468
25	0.48	446.55	00:00:02.4070000	17550	0.09	0.00	00:11:33.8665315	16988049
30	0.32	377.05	00:00:02.4630000	21060	0.07	0.00	00:16:33.5639800	27455571
35	0.23	251.52	00:00:02.6800000	24570	0.06	0.00	00:23:29.2769294	45186687
40	0.19	202.08	00:00:02.7630000	28080	0.06	0.00	00:30:40.8971535	60633846
45	0.11	78.37	00:00:02.8000000	31590	0.06	0.00	00:39:59.4908467	86223852
50	0.16	155.87	00:00:02.9300000	35100	0.06	0.00	00:48:49.4123436	106847559

Table 9. Results for dataset DIABETS.

k	FCM (average of 10 times running)				FGMCM			
	f	E	t	D_{ave}	f	E	t	D_{ave}
2	27659.72	17.51	00:00:04.1670000	3072	23538.96	0.00	00:00:07.2175795	3072
3	9763.50	0.37	00:00:04.6100000	149068.8	9727.04	0.00	00:00:16.5557688	79104
4	5487.53	0.00	00:00:05.3130000	538521.6	5487.53	0.00	00:00:26.1325551	678144
5	3455.16	0.18	00:00:05.8700000	619776	3448.99	0.00	00:00:39.6741339	1000704
6	2359.75	0.00	00:00:05.9230000	814694.4	2359.70	0.00	00:00:58.4543885	2341632
7	1992.64	13.02	00:00:08.7770000	1237555.2	1763.07	0.00	00:01:19.9486974	3782400
8	1318.21	0.00	00:00:08.7370000	1731993.6	1318.21	0.00	00:01:46.6716950	6381312
9	1048.69	2.40	00:00:08.4670000	1774310.4	1024.11	0.00	00:02:15.4989234	8648448
10	866.82	10.02	00:00:09.6730000	1499136	787.85	0.00	00:02:44.2900472	10184448
11	710.54	4.16	00:00:10.7700000	2257305.6	682.19	0.00	00:03:22.6204079	14408448
12	588.96	2.34	00:00:10.3570000	2547302.4	575.50	0.00	00:03:54.6121772	14804736
13	531.67	4.30	00:00:11.2700000	2432102.4	509.77	0.00	00:04:33.5251038	18099456
14	464.47	1.24	00:00:13.8970000	3114854.4	458.80	0.00	00:05:10.9153024	18572544
15	391.31	2.15	00:00:10.4730000	2287872	383.06	0.00	00:05:55.2400818	21314304
20	215.37	0.00	00:00:21.6330000	6248448	229.15	6.40	00:10:30.6439724	42306048
25	133.48	0.00	00:00:26.3530000	7981440	148.92	11.56	00:16:43.5791164	77933568
30	83.12	0.00	00:00:22.0000000	6465024	113.66	36.73	00:23:36.1694973	105924096
35	51.83	0.00	00:00:20.8870000	5693184	89.01	71.74	00:31:41.5688671	136035072
40	30.39	0.00	00:00:26.2900000	7928832	57.61	89.57	00:40:03.2513109	149618688
45	10.58	59.94	00:00:17.3500000	4064256	6.61	0.00	00:49:30.0585452	157671168
50	1.83	95.69	00:00:13.2730000	1693440	0.94	0.00	00:59:55.9337937	168603648

These results for medium size datasets demonstrate that the proposed algorithm finds also better or very similar solutions for many times in comparison with average FCM algorithm. This algorithm fails only for big number of clusters *k* on WINE and DIABETS datasets. But gives always good solutions for CLEVELAND, LIVER, IONOSPHERE datasets. Although it is needed more CPU computation time for proposed algorithm according to compution operations, it can be used to get more effective solutions.

Results for large size datasets are presented in tables 10, 11, 12.

Table 10. Results for dataset TSP.

k	FCM (average of 10 times running)				FGMCM			
	f	E	t	D_{ave}	f	E	t	D_{ave}
2	$97.19 * 10^{14}$	19.56	00:00:06.1130000	5300	$81.29 * 10^{14}$	0.00	00:00:04.0158374	8480
3	$46.65 * 10^{14}$	0.48	00:00:05.9170000	198750	$46.43 * 10^{14}$	0.00	00:00:16.3566228	145220
4	$31.60 * 10^{14}$	1.90	00:00:06.0170000	256520	$31.01 * 10^{14}$	0.00	00:00:22.8667654	509860
5	$23.01 * 10^{14}$	0.20	00:00:10.3300000	601020	$22.96 * 10^{14}$	0.00	00:00:24.6412358	780160
6	$18.21 * 10^{14}$	0.68	00:00:07.7300000	587028	$18.09 * 10^{14}$	0.00	00:00:34.2224419	1079080
7	$14.81 * 10^{14}$	0.00	00:00:06.4770000	454104	$15.85 * 10^{14}$	7.05	00:00:43.1700302	1093920
8	$12.16 * 10^{14}$	1.05	00:00:10.5770000	919232	$12.03 * 10^{14}$	0.00	00:00:49.4258591	1687520
9	$10.70 * 10^{14}$	3.60	00:00:07.1370000	776556	$10.33 * 10^{14}$	0.00	00:01:06.7601022	2393480
10	$9.42 * 10^{14}$	5.94	00:00:08.1330000	1555020	$8.89 * 10^{14}$	0.00	00:01:18.3914718	3061280
11	$7.93 * 10^{14}$	1.20	00:00:08.2370000	1449338	$7.83 * 10^{14}$	0.00	00:02:22.6365732	4472140
12	$7.10 * 10^{14}$	1.13	00:00:07.7800000	1235112	$7.02 * 10^{14}$	0.00	00:02:21.8094964	5985820
13	$6.35 * 10^{14}$	0.00	00:00:08.3100000	1588834	$6.48 * 10^{14}$	2.02	00:02:48.5704486	6123620
14	$5.68 * 10^{14}$	0.00	00:00:10.8930000	2916060	$5.69 * 10^{14}$	0.26	00:03:07.8699913	8572220
15	$5.23 * 10^{14}$	0.00	00:00:09.0870000	2012940	$5.23 * 10^{14}$	0.02	00:02:46.2050137	10639220
20	$3.65 * 10^{14}$	2.65	00:00:10.8270000	3014640	$3.56 * 10^{14}$	0.00	00:04:48.1725781	26112040
25	$2.69 * 10^{14}$	0.00	00:00:15.4200000	5936000	$2.71 * 10^{14}$	0.85	00:06:53.4707062	57902500
30	$2.14 * 10^{14}$	0.00	00:00:14.9930000	5440980	$2.20 * 10^{14}$	2.74	00:09:23.3998918	74704560
35	$1.70 * 10^{14}$	0.00	00:00:19.8470000	8614620	$1.77 * 10^{14}$	4.00	00:12:33.1384653	121143160
40	$1.47 * 10^{14}$	0.00	00:00:18.6770000	7844000	$1.48 * 10^{14}$	1.24	00:15:55.8313359	137049520
45	$1.25 * 10^{14}$	0.00	00:00:16.4770000	6057900	$1.31 * 10^{14}$	4.22	00:20:46.8459217	194088120
50	$1.08 * 10^{14}$	0.00	00:00:21.9500000	9736100	$1.20 * 10^{14}$	11.08	00:23:52.5072863	232558700

Table 11. PAGE Results for dataset BLOCK.

k	FCM (average of 10 times running)				FGMCM			
	f	E	t	D_{ave}	f	E	t	D_{ave}
2	$71.34 * 10^6$	58.51	00:00:05.6570000	21892	$45.01 * 10^6$	0.00	00:05:38.5083680	21892
3	$26.20 * 10^6$	5.58	00:00:07.7530000	666611.4	$24.82 * 10^6$	0.00	00:14:22.9942258	4356508
4	$14.70 * 10^6$	0.00	00:00:08.4230000	849409.6	$15.70 * 10^6$	6.95	00:24:54.6211853	11537084
5	$10.20 * 10^6$	16.61	00:00:09.9600000	1526967	$8.74 * 10^6$	0.00	00:38:29.0036795	18405699
6	$7.57 * 10^6$	20.22	00:00:11.4900000	2026104.6	$6.29 * 10^6$	0.00	00:53:55.8814841	25236003
7	$6.35 * 10^6$	24.17	00:00:13.3400000	2654952.3	$5.12 * 10^6$	0.00	01:13:28.3586868	44391503
8	$4.34 * 10^6$	28.57	00:00:11.4700000	1983415.2	$3.38 * 10^6$	0.00	01:35:12.7572140	66283503
9	$3.66 * 10^6$	36.64	00:00:14.5870000	3088413.9	$2.68 * 10^6$	0.00	01:58:17.4563351	90912003
10	$2.69 * 10^6$	24.12	00:00:18.9670000	4641104	$2.17 * 10^6$	0.00	02:23:59.2562800	106510053
11	$2.03 * 10^6$	12.32	00:00:25.8730000	7200278.8	$1.81 * 10^6$	0.00	02:52:30.4134416	117948623
12	$2.03 * 10^6$	34.02	00:00:25.0570000	6928818	$1.52 * 10^6$	0.00	03:23:12.9475463	147896879

Table 12. Results for dataset PENDIGIT.

k	FCM (average of 10 times running)				FGMCM			
	f	E	t	D_{ave}	f	E	t	D_{ave}
2	$228.11 * 10^4$	26.02	00:00:11.7670000	43968	$181.01 * 10^4$	0.00	00:09:23.5020858	43968
3	$56.72 * 10^4$	0.00	00:00:15.3000000	890352	$57.05 * 10^4$	0.59	00:23:57.9966188	241824
4	$31.40 * 10^4$	1.53	00:00:16.9430000	1336627.2	$30.92 * 10^4$	0.00	00:42:33.2228959	8551776
5	$17.80 * 10^4$	2.75	00:00:27.7370000	4105512	$17.31 * 10^4$	0.00	01:04:21.6022562	13278336
6	$12.80 * 10^4$	0.00	00:00:32.9870000	5427849.6	$13.08 * 10^4$	1.81	01:31:24.1976474	13542144
7	$8.27 * 10^4$	0.95	00:00:25.8970000	3777950.4	$8.20 * 10^4$	0.00	02:03:20.1091332	20082384
8	$6.39 * 10^4$	2.98	00:00:33.7400000	5751014.4	$6.21 * 10^4$	0.00	02:39:49.8736324	21489360

On TSP dataset the proposed algorithm gives 11 times better, 8 times very similar (with Error $\leq$ 5%) and 2 times worth solutions. On BLOCK and PENDIGIT datasets proposed algotithm gives better solutions for small number of clusters *k*.

Results represented in Table 2-12 demonstrate that average calculation time for FCM running 10 times are faster for large datasets with big cluster count, but the aim function value for FGMCM is better than average FCM running 10 times for small, medium and large datasets.

6. Conclusion

In this paper, we have developed a new algorithm for solving fuzzy clustering problems. We used nonsmooth nonconvex formulation of the hard clustering problem and obtained an incremental algorithm for solving of the fuzzy clustering problem. Such an approach allows us to use powerful incremental algorithms in hard clustering for solving of fuzzy clustering problem. We presented the results of numerical experiments using 11 real-world datasets and compared the proposed algorithm with fuzzy k-means algorithm. The results of numerical experiments clearly demonstrate that the proposed algorithm is more accurate than the fuzzy k-means in finding global minimizers of the clustering function.

References

[1] Zadeh, A.L., Fuzzy sets, **Information and Control**, 8, 338-353, (1965).

[2] Al-Sultan, K.S., A tabu search approach to the clustering problem, **Pattern Recognition**, 28(9), 1443-1451, (1995).

[3] Brown, D.E., Entail, C.L., A practical application of simulated annealing to the clustering problem, **Pattern Recognition**, 25, 401-412, (1992).

[4] Diehr, G., Evaluation of a branch and bound algorithm for clustering, **SIAM J. Scientific and Statistical Computing**, 6, 268-284, (1985).

[5] Dubes, R., Jain, A.K., Clustering techniques: the user's dilemma, **Pattern Recognition**, 8, 247-260, (1976).

[6] Hanjoul, P., Peeters, D., A comparison of two dual-based procedures for solving the p-median problem, **European Journal of Operational Research**, 20, 387-396, (1985).

[7] Hansen, P., Jaumard, B., Cluster analysis and mathematical programming, **Mathematical Programming**, 79(1-3), 191-215, (1997).

[8] Hansen, P., Mladenovic, N., J-means: a new heuristic for minimum sum-of-squares clustering, **Pattern Recognition**, 4, 405-413, (2001).

[9] Hansen, P., Mladenovic, N., Variable neighborhood decomposition search, **Journal of Heuristic**, 7, 335-350, (2001).

[10] Koontz, W.L.G., Narendra, P.M., Fukunaga, K., A branch and bound clustering algorithm, **IEEE Transactions on Computers**, 24, 908-915, (1975).

[11] Du Merle, O., Hansen, P., Jaumard, B., Mladenovic, N., An interior point method for minimum sum-of-squares clustering, **SIAM Journal on Scientific Computing**, 21, 1485-1505, (2001).

[12] Selim, S.Z., Al-Sultan, K.S., A simulated annealing algorithm for the clustering, **Pattern Recognition**, 24(10), 1003-1008, (1991).

[13] Spath, H., **Cluster Analysis Algorithms**, Ellis Horwood Limited, Chichester, (1980).

[14] Sun, L.X., Xie, Y.L., Song, X.H., Wang, J.H., Yu, R.Q., Cluster analysis by simulated annealing, **Computers and Chemistry**, 18, 103-108, (1994).

[15] Likas, A., Vlassis, N. and Verbeek, J., The global k-means clustering algorithm, **Pattern Recognition**, 36, 451 – 461, (2001).

[16] Bagirov, A.M., Modified global k-means algorithm for minimum sum-of-squares clustering problems, **Pattern Recognition**, 41, 3192- 3199, (2008).

[17] Vanisri, D. and Loganathan, C., An efficient fuzzy clustering algorithm based on modified k-means, **International Journal of Engineering Science and Technology**, 5949-5958, (2010).

[18] Ordin, B., Bagirov, A., A heuristic algorithm for solving the minimum sum of squares clustering problems, **Journal of Global Optimization**, 61, 341–361, (2015).

[19] Bagirov, A., Ordin, B., Ozturk, G., Xavier, A.E., An ıncremental clustering algorithm based on hyperbolic smoothing, Computational Optimization and Applications, 61(1), 219-241, (2015).

[20] Chuang, K., Tzeng, H., Chen, S., Wu, J. and Chen, T., Fuzzy c-means clustering with spatial information for image segmentation, **Computerized Medical Imaging and Graphics**, 30, 9–15, (2006).

[21] Brandt, M.E., Bohant, T.P., Kramer, L.A. and Fletcher, J.M., Estimation of CSF, white and gray matter volumes in hydrocephalic children using fuzzy clustering of Mr Images, **Computerized Medical Imaging and Graphics**, 18(1), 25–34, (2004).

[22] Staianoa, A., Tagliaferria, R. and Pedrycz, W., Improving RBF networks performance in regression tasks by means of a supervised fuzzy clustering, **Neurocomputing**, 69, 1570-1581, (2005).

[23] Velmurugan, T. and Santhanam, T., Performance evaluation of k-means and fuzzy c-means clustering algorithms for statistical distributions of input data points, **European Journal of Scientific Research**, 3, 320-330, (2010).

[24] Bagirov, A.M., Yearwood, J., A new nonsmooth optimization algorithm for minimum sum-of-squares clustering problems, **European Journal of Operational Research**, 170(2), 578–596, (2006).

[25] Bezdek, J.C., Ehrlich, R., Full, W., FCM: The fuzzy c-means clustering algorithm, **Computers & Geosciences**, 10(2–3), 191–203, (1984).

[26] UC Irvine Machine Learning Repository (http://archive.ics.uci.edu/ml/)