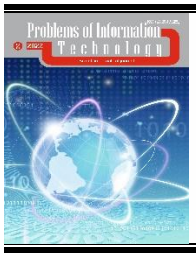


Available online at www.jpit.az

13 (2)
2022

AN ADAPTIVE k-MEDIANS CLUSTERING ALGORITHM

Adil M. Bagirov [1,a], Sona Taheri [2,a,b], Burak Ordin [3,c]

[a] School of Engineering, Information Technology and Physical Sciences, Federation University Australia, Ballarat, Australia

[b] School of Mathematical Sciences, RMIT University, Melbourne, Australia

[c] Department of Mathematics, Faculty of Science, Ege University, Izmir, Turkey

[1] a.bagirov@federation.edu.au; [2] sona.taheri@rmit.edu.au; [3] burak.ordin@ege.edu.tr

[1] 0000-0003-2075-1699; [2] 0000-0003-1779-4567; [3] 0000-0001-7897-3265

ARTICLE INFO

http://doi.org/10.25045/jpit.v13.i2.01

Article history:
Received 14 March 2022
Received in revised form 27 May 2022
Accepted 17 June 2022

Keywords:
Cluster analysis
k-medians algorithm
Adaptive clustering

ABSTRACT

A new version of the k-medians algorithm, the adaptive k-medians algorithm, is introduced to solve clustering problems with the similarity measure defined using the L_1-norm. The proposed algorithm first calculates the center of the whole data set as its median. To solve the k-clustering problem ($k > 1$), we formulate the auxiliary clustering problem to generate a set of starting points for the k-th cluster center. Then, the k-medians algorithm is applied starting from the previous $(k-1)$ cluster centers and each point from the set of starting points to solve the k-clustering problem. A solution with the least value of the clustering function is accepted as the solution to the k-clustering problem. We evaluate the performance of the adaptive k-medians algorithm and compare it with other concurrent clustering algorithms using 8 real-world data sets.

1. Introduction

Clustering deals with the problem of organizing a collection of objects into clusters based on their similarity. It has a wide range of applications in many fields (Bagirov, Karmitsa, & Taheri, 2020; Castellanos, Cigarran, & Garcia-Serrano, 2017; Dai et al., 2019; Jain, 2010). The similarity measure is a fundamental notion in cluster analysis. In data sets with only numeric attributes this measure can be defined using different norms. Clustering problems with the similarity measure defined using the squared Euclidean norm have been studied in far more detailed than those with similarity measures based on other norms (Bagirov, Karmitsa & Taheri, 2020; Bai et al., 2013; Karmitsa, Bagirov & Taheri, 2017; Karmitsa, Bagirov & Taheri, 2018; Lai & Huang, 2010; Xavier & Xavier, 2011).

Clustering algorithms with the similarity measure defined using the L_1-norm are more robust to outliers than those with the squared Euclidean norm (Zhang et al., 2012). These algorithms are more preferable in high dimensional data (Aggarwal, Hinneburg, & Keim, 2001). They produce the highest identification and the best verification rates in the speaker recognition systems (Hanilci & Ertas, 2011).

To the best of our knowledge, the paper (Carmichael & Sneath, 1969) is the first where the clustering problem with the L_1-norm is considered. The k-medians algorithm was developed in (Spath, 1976). The ISODATA algorithm with the L_1-norm was introduced in

(Jajuga, 1987). In the paper (de Souza R & de Carvalho, 2004), the authors introduce adaptive and non-adaptive clustering algorithms using the L_1- norm.

The one-dimensional center-based L_1-clustering algorithm is proposed in (Sabo, Scitovski, & Vazler, 2013). The incremental algorithms, where clustering functions defined using the L_1-norm are approximated by smooth functions, are developed in (Bagirov & Mohebi, 2016; Bagirov & Taheri, 2017). A nonsmoothed optimization-based clustering algorithm is introduced in (Bagirov, Ordin, & Mohebi, 2020).

The k-medians algorithm is a fast and easy to implement clustering algorithm. It has been applied in different areas including detecting outliers in noisy data (Friggstad et al., 2019) and the local improvement of traveling salesman tours (Khachay & Khachay, 2019). This algorithm is sensitive to the choice of starting points and finds only local solutions that can be significantly different from global solutions in large data sets. In addition, solutions found by this algorithm deteriorate considerably as the size of a data set increases.

An adaptive k-medians algorithm, proposed in this paper, overcomes these difficulties. The design of this algorithm is similar to that of the modified global k-means algorithm (Bagirov, 2008). However, its properties and performance are different, and their study are worthwhile. The algorithm starts with the calculation of the center of the whole data set and gradually adds new cluster centers. Using an auxiliary clustering problem, a special procedure is used to generate starting cluster centers. The k-medians algorithm is applied to solve both the clustering and the auxiliary clustering problems. Using computational results on 8 real-world data sets, we analyze the performance of the proposed algorithm and compare it with other similar clustering algorithms.

The rest of the paper is organized as follows. The descriptions of the k-medians and the partial k-medians algorithms are given in Section 2. An algorithm for generating initial cluster centers is described in Section 3. An adaptive k-medians algorithm is introduced in Section 4. Computational results are reported in Section 5, and Section 6 presents conclusions and discussions.

2. k-medians algorithms

We start by formulating the clustering and the auxiliary clustering problems.

2.1. Clustering and auxiliary clustering problems

Assume that a finite set $A = \{a^1, \dots, a^m\} \subset \mathbb{R}^n$ is given, where a^i are data points with n attributes. The hard clustering problem is the distribution of the points of the set A into a given number k of nonempty and pairwise disjoint subsets (clusters) A^j, $j \in J_k = \{1, \dots, k\}$ such that their union is the set A. Each cluster A^j is represented by its center $x^j \in \mathbb{R}^n, j \in J_k$. We define the similarity measure as the distance function d_1 defined using the L_1-norm.

The nonconvex nonsmooth optimization model of the clustering problem is (Bagirov & Yearwood, 2006):

$$\begin{cases} \text{minimize} & f_k(x) \\ \text{subject to} & x = (x^1, \dots, x^k) \in \mathbb{R}^{nk} \end{cases} \quad (1)$$

where

$$f_k(x^1, \dots, x^k) = \frac{1}{m} \sum_{a \in A} \min_{j \in J_k} d_1(x^j, a). \quad (2)$$

The function f_k is called the *k-th cluster function* and the problem (1) is called *the k-clustering (k-partition) problem* (Bagirov & Yearwood, 2006). The problem (1) is also called *the minimum sum-of-absolutes clustering (MSAC) problem*.

Let $x^1, \dots, x^{l-1}, l > 1$ be a solution to the $(l-1)$-clustering problem. The function

$$\bar{f}_l(y) = \frac{1}{m} \sum_{a \in A} \min\{r^a_{l-1}, d_1(y, a)\}, \; y \in \mathbb{R}^n \quad (3)$$

is called *l-th auxiliary cluster function* (Ordin & Bagirov, 2015). Here,

$$r^a_{l-1} = \min_{j \in J_{l-1}} d_1(x^j, a), a \in A. \quad (4)$$

The *l-th auxiliary clustering problem* is

$$\begin{cases} \text{minimize} & \bar{f}_l(y) \\ \text{subject to} & y \in \mathbb{R}^n. \end{cases} \quad (5)$$

Figure 1 illustrates the auxiliary clustering function for $l = 3$ in a data set containing 18 points from $\mathbb{R}$. We compute the numbers r^a_2 and formulate the function $\bar{f}_3(y)$. This function has

three local minimizers and there are regions where it is a constant. These regions contain points which are away from data instances.

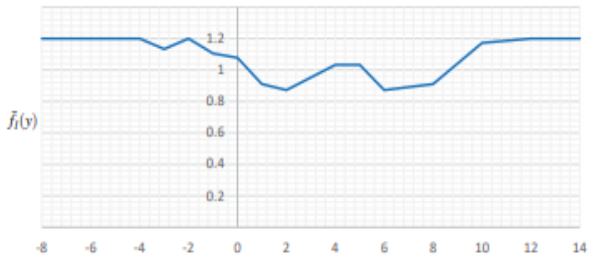


Fig. 1. Graph of the auxiliary cluster function $\bar{f}_3(y)$.

2.2. The k-medians and partial k-medians algorithms

In order to solve the k-clustering problem the k-medians algorithm (Spath, 1976):

Step 1. Selects (randomly) k initial cluster centers $x^1, \dots, x^k$;

Step 2. Allocates each data point to the closest cluster center using the distance function d_1 and finds the cluster partition $A^1, \dots, A^k$;

Step 3. Updates cluster centers $x^1, \dots, x^k$; as vectors of medians of attributes using data points from the clusters $A^1, \dots, A^k$ respectively;

Step 4. Repeats Steps 2 and 3 until no data point changes its cluster.

Note that in Step 3, the center of each cluster is calculated. For a cluster C, the problem of finding its center can be formulated as

$$\begin{cases} \text{minimize} & \varphi(y) = \frac{1}{|C|} \sum_{c \in C} d_1(y, c) \\ \text{subject to} & y \in \mathbb{R}^n, \end{cases} \qquad (6)$$

where $|C|$ is the cardinality of the set C. It is known that the coordinates of the solution y to this problem are medians of the corresponding attributes (Bagirov, Karmitsa, & Taheri, 2020; Bagirov, Ordin, & Mohebi, 2020; Spath, 1976).

Next, we describe an algorithm for solving the auxiliary clustering problem (5). Let $\tilde{f}_{l-1}(y) = \frac{1}{m} \sum_{a \in A} r_{l-1}^a$ be the optimal value of the $(l-1)$-clustering problem. Following (Ordin & Bagirov, 2015) consider the sets:

$$S_1 = \{y \in \mathbb{R}^n : d_1(y, a) \geq r_{l-1}^a \ \forall a \in A\}, \text{ and}$$

$$S_2 = \left\{ y \in \mathbb{R}^n : \exists\, \bar{A} \subset A,\ \bar{A} \neq \emptyset : d_1(y, a) < r_{l-1}^a \ \forall a \in \bar{A} \right\}$$

Note that $S_1 \cap S_2 = \emptyset$ and $S_1 \cup S_2 = \mathbb{R}^n$. Starting points for solving the problem (5) are chosen from the set S_2 as only these points provide the value of $\bar{f}_l$ less than the optimal value $\tilde{f}_{l-1}$ of the $(l-1)$-clustering problem. Figure 2 illustrates S_1 and S_2.

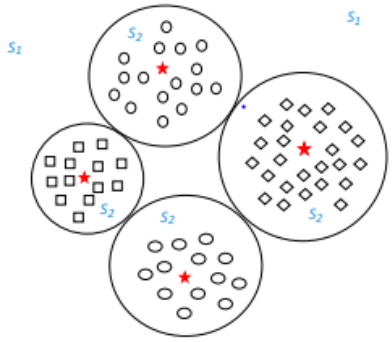


Fig. 2. Illustration of sets S_1 and S_2.

There are four clusters in this figure. Their centers are shown by "red" stars. The set S_2 consists of all points inside four balls except cluster centers and the set S_1 contains the centers of clusters and the part of the space outside balls.

A version of the k-medians algorithm for solving the auxiliary clustering problem (5) is given in Algorithm 1, where all centers, but the l-th center, are fixed. Therefore, we call this version *the partial k-medians algorithm*.

Note that the stopping criterion in Step 2 means that the algorithm terminates when no data point moves out or in of the new cluster. Since for all points $y^q \in S_2$ the sets $B(y^q)$, defined in (7), are not empty the problem of finding its center in Step 3 is well defined. The center $\bar{y}_c$ is computed as a solution to the problem (6) for $C = B(y^q)$.

Algorithm 1. Partial k-medians algorithm.

Input. Data set A and the solution $(x^1, \dots, x^{l-1}) \in \mathbb{R}^{n(l-1)}$ to the $(l-1)$-partition problem, $l > 1$.

Output. Solution $\bar{y} \in \mathbb{R}^n$ to l-th auxiliary clustering problem.

Step 0. *(Initialization)* Select a starting point $y^1 \in S_2$. Set iteration counter $q = 1$.

Step 1. Compute the set

$$B(y^q) = \{a \in A : d_1(y^q, a) < r_{l-1}^a\}. \qquad (7)$$

Step 2. *(Stopping criterion)* If $B(y^q) = B(y^{q-1})$ for $q > 1$, then the algorithm terminates with the solution $\bar{y} = y^q$ to the auxiliary clustering problem.

Step 3. Find a center $\bar{y}_c$ of the set $B(y^q)$ by computing its coordinates as the medians of the

corresponding attributes.

Step 4. Set $y^{q+1} = \bar{y}_c, q = q + 1$ and go to Step 1.

3. Computation of initial cluster centers

The k-medians algorithm is a local search algorithm and its success in finding high quality clustering solutions heavily depends on the choice of initial points. In this section, we describe an algorithm for generating initial points which are "rough" solutions to the clustering problem. The similar procedure was introduced in (Ordin & Bagirov, 2015).

Take any $y \in S_2$ and divide the set A into the following two subsets:

$$\bar{B}_1(y) = \{a \in A: d_1(y,a) \geq r_{l-1}^a\} \text{ and}$$

$$\bar{B}_2(y) = \{a \in A: d_1(y,a) < r_{l-1}^a\}.$$

The set $\bar{B}_2(y)$ contains all data points $a \in A$ which are closer to the point y than to their cluster centers, and the set $\bar{B}_1(y)$ includes all other data points. Since $y \in S_2$ the set $\bar{B}_2(y) \neq \emptyset$. Furthermore, $\bar{B}_1(y) \cap \bar{B}_2(y) = \emptyset$ and $A = \bar{B}_1(y) \cup \bar{B}_2(y)$. Figure 3 depicts the sets $\bar{B}_1(y)$ and $\bar{B}_2(y)$ for a given y (black ball). There are three clusters in this data set and their centers are shown by "red" stars. The set $\bar{B}_2(y)$ contains all "yellow" data points and the set $\bar{B}_1(y)$ all other points.

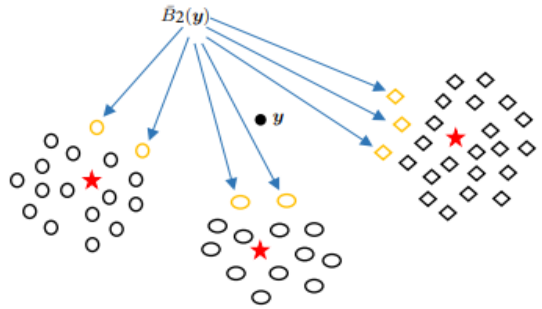


Fig. 3. Illustration of sets $\bar{B}_1(y)$ and $\bar{B}_2(y)$.

An algorithm for finding starting points for solving the auxiliary clustering problem (5) is presented in Algorithm 2.

Algorithm 2. Finding the set of initial points for the auxiliary clustering problem.

Input. The solution $(x^1, \dots, x^{l-1}) \in \mathbb{R}^{n(l-1)}$ to the $(l-1)$-clustering problem, $l > 1$ and numbers $\gamma_1, \gamma_2 \in [0,1]$.

Output. The set $\bar{S}$ of initial points for the l-th cluster center.

Step 1. For each $a \in A \backslash S_1$, compute

$$z_l(a) = \frac{1}{m}\sum_{b \in A} \max\{0, r_{l-1}^b - d_1(a,b)\}. \quad (8)$$

Step 2. Compute

$$z_{1max} = \max_{a \in A \backslash S_1} z_l(a) \quad (9)$$

and the set $\bar{A}_1 = \{a \in A \backslash S_1: z_l(a) \geq \gamma_1 z_{1max}\}$.

Step 3. For each $a \in \bar{A}_1$ compute the set $\bar{B}_2(a)$ and its center $c(a)$. Denote by $\bar{A}_2$ the set of all such centers.

Step 4. For each $c \in \bar{A}_2$ compute $z_{2l}(c) = z_l(c)$ using (8), then compute

$$z_{2max} = \max_{c \in \bar{A}_2} z_{2l}(c), \quad (10)$$

and the set $\bar{A}_3 = \{c \in \bar{A}_2: z_{2l}(c) \geq \gamma_2 z_{2max}\}$. Set $\bar{S} = \bar{A}_3$ and **STOP**.

Remark 1. The number $z_l(a)$ given in (8) shows the decrease of the value of the l-th cluster function f_l comparing with the value $f_{l-1}(x^1, \dots, x^{l-1})$ if points $(x^1, \dots, x^{l-1}, a)$ are chosen as cluster centers for the l-clustering problem. The number z_{1max}, defined in (9), is the maximum decrease among all points of the set A, and the number z_{2max}, given in (10), shows the largest value of decrease among all centers $c \in \bar{A}_2$.

Remark 2. Note that the set of starting points for solving the auxiliary clustering problem (5) is computed using data points from the set A. The use of the parameters γ_j, $j = 1,2$ enables us to define thresholds which in turn allow to keep only those points providing the significant decrease of the clustering function in comparison with its value at the previous iteration. The set $\bar{A}_3$ contains all such points which are used as starting points for solving the auxiliary clustering problem (5).

Next, we present an algorithm for finding starting points for solving the clustering problem (1). This algorithm in its turn applies Algorithms 1 and 2 to find solutions of the auxiliary clustering problem (5). The lowest value of the auxiliary function over these solutions are computed and a subset of the solutions is defined using this value and a predefined threshold. The points from this subset are considered as initial points for solving the clustering problem (1).

4. Adaptive k-medians clustering algorithm

In this section, we introduce an adaptive k-medians (AkMed) algorithm for solving the clustering problem (1). We also study its time complexity and discuss different approaches to reduce it. Similar algorithms for solving clustering problems with L_2-norm are developed, for example, in (Bagirov & Yearwood, 2006; Likas, Vlassis, & Verbeek, 2003; Ordin & Bagirov, 2015).

4.1. AkMed *algorithm*

AkMed given in Algorithm 4, solves all intermediate l-clustering problems, $l \in J_{k-1}$, in addition to the k-clustering ($k > 1$) problem.

Algorithm 3 Finding the set of initial points for the clustering problem.

Input. The solution $(x^1, \dots, x^{l-1}) \in \mathbb{R}^{n(l-1)}$ to the $(l-1)$-clustering problem, $l > 1$ and numbers $\gamma_1, \gamma_2 \in [0,1]$ and $\gamma_3 \in [1, \infty)$.

Output. The set S_0^l of initial cluster centers for the l-clustering problem.

Step 1. Apply Algorithm 2 to compute the set $\bar{A}_3$ of starting points for solving the l-th auxiliary clustering problem (5).

Step 2. Apply Algorithm 1 to compute the set $\bar{A}_4$ of local minimizers of the auxiliary clustering problem (5) using starting points from the set $\bar{A}_3$.

Step 3. Compute the number $\bar{f}_l^{min} = \min\limits_{y \in \bar{A}_4} \bar{f}_l(y)$.

Step 4. Compute the set $\bar{A}_5 = \{y \in \bar{A}_4 : \bar{f}_l(y) \leq \gamma_3 \bar{f}_l^{min}\}$.

Step 5. Construct the set $S_0^l = \{(x^1, \dots, x^{l-1}, y) : y \in \bar{A}_5\}$ and **STOP**.

4.2. Time complexity of the AkMed algorithm

In this subsection we estimate the time complexity of the AkMed algorithm. Such estimations for different versions of the k-medians algorithm can be found, for example, in (Ben-David, 2020; Kumar, Sabharwal, & Sen, 2010) and the estimation of the accuracy of one version of this algorithm is presented in (Khachay & Khachay, 2019).

The AkMed algorithm involves the calculation of cluster centers, distances between data points and also distances between data points and cluster centers. This algorithm starts with the calculation of the center of the whole data set and gradually adds one cluster at each iteration. The total number of iterations by the k-medians algorithm is k. The k-medians algorithm and Algorithms 1, 2 and 3 are applied at each iteration. Next we describe the time complexity of each of these algorithms.

The time complexity of one distance function evaluation is $O(n)$ where n is the number of input variables (attributes). The k-medians algorithm requires $O(ml)$ distance function evaluations at the l-th iteration, $1 \leq l \leq k$. The complexity of an algorithm for finding medians in the worst case can be estimated as $O(m^2)$, and therefore, the time complexity of one iteration by the k-medians algorithm can be estimated as $O(mn(l+m))$.

Algorithm 4. The AkMed algorithm.

Input. Data set A and the number of clusters k to be computed.

Output. Solutions $(x^1, \dots, x^l) \in \mathbb{R}^{nl}$ to the l-clustering problem, $l = 1, \dots, k$.

Step 1. *(Initialization)* Compute the center $x^1 \in \mathbb{R}^n$ of the set A. Set $l = 1$.

Step 2. *(Stopping criterion)* Set $l = l + 1$. If $> k$, then **STOP**. The k-clustering problem has been solved.

Step 3. *(Computation of a set of initial points)* Apply Algorithm 3 to compute the set S_0^l of initial cluster centers.

Step 4. *(Computation of a set of cluster centers)* Take each $(y^1, \dots, y^l) \in S_0^l$ as starting cluster centers, apply the k-medians algorithm to solve the clustering problem (1) and find a solution $(\hat{y}^1, \dots, \hat{y}^l)$. Let $\bar{A}_6$ be a set of all solutions obtained using starting cluster centers from the set S_0^l.

Step 5. *(Computation of the best solution)* Compute $f_l^{min} = \min\{f_l(\hat{y}^1, \dots, \hat{y}^l) : (\hat{y}^1, \dots, \hat{y}^l) \in \bar{A}_6\}$, and the collection of centers $(\bar{y}^1, \dots, \bar{y}^l)$ such that $f_l(\bar{y}^1, \dots, \bar{y}^l) = f_l^{min}$.

Step 6. *(Solution to the l-partition problem)* Set $(x^j = \bar{y}^j, j = 1, \dots, l$ as a solution to the l-partition problem and go to Step 2.

Since this algorithm is applied starting from different points and the number of these points is no greater than the total number of points (m) in a data set we conclude that the time complexity of the k-medians algorithm is $O(nm^2(l+m))$. Let T_1 be the maximum number of iterations by the k-medians algorithm in one iteration of Algorithm 4. Then the total time complexity of the k-medians algorithm in one iteration of Algorithm 4 is $O(T_1 nm^2(l+m))$.

Each iteration of Algorithm 1 requires $O(m)$ distance function evaluations. Each iteration also includes the median finding problem, and therefore, the time complexity of one iteration by this algorithm can be estimated as $O(nm^2)$. Algorithm 1 is applied starting from different points where the number of these points is no greater than m, and thus, the time complexity of Algorithm 1 is $O(nm^3)$. Let T_2 be a maximum number of iterations by this algorithm in one iteration of Algorithm 4. Then the total time complexity of Algorithm 1 in one iteration of Algorithm 4 is $O(T_2 nm^3)$.

Algorithm 2 requires $O(m)$ distance function evaluations and in the worst case $O(m^2)$ operations to compute a data point with the maximum decrease of the cluster function. Then, the time complexity of this algorithm at each iteration of Algorithm 4 can be estimated as $O(m(n+m))$.

Algorithm 3 contains only minimum operation, and therefore, the time complexity of this algorithm at each iteration of Algorithm 4 is $O(m^2)$. Summarizing all described above, we conclude that the time complexity of Algorithm 4 can be expressed as $O(knm^3T)$, where $T = \max\{T_1, T_2\}$.

4.3. Reduction of complexity

In this subsection, we discuss three different approaches to reduce the computational effort required by the AkMed algorithm.

Reduction of the number of starting cluster centers. Starting cluster centers are computed in Algorithms 2 and 3. The initial set of such centers is defined as $A\backslash S_1$ in Step 1 of Algorithm 2. At the l-th iteration ($l \geq 2$) of the adaptive algorithm one can remove points from the set $A\backslash S_1$ which are close to cluster centers $x^1, \ldots, x^{l-1}$. We use the following scheme for this purpose. For each cluster $A^q, q \in J_{l-1}$ compute its average radius $R_{av}^q = \frac{1}{|A^q|}\sum_{a\epsilon A^q} d_1(x^q, a)$, and define the subset $\hat{A}^q = \{a \in A^q : d_1(x^q, a) \geq R_{av}^q\}$. Note that $\hat{A}^q \neq \emptyset$ under the assumption that $A^q \neq \emptyset$. Consider the set $\hat{A} = \bigcup_{q=1}^{l-1} \hat{A}^q$. It is clear that the set $\hat{A}$ is a subset of A. Replacing the set $A\backslash S_1$ by the set $\hat{A}\backslash S_1$ in Step 1 of Algorithm 2 leads to the reduction of the number of starting cluster centers (in some cases significantly) since the points that do not provide sufficient decrease of the cluster function are removed.

Exclusion of close stationary points. Algorithm 1 solves the auxiliary clustering problems using starting points generated by Algorithm 2. This is done in Step 2 of Algorithm 3. Since the partial k-medians algorithm may find the same (or close) local solutions to the auxiliary clustering problem starting from completely different points we can remove such points from the set $\bar{A}_4$ using a tolerance $\varepsilon > 0$. Define $\varepsilon = \hat{f}_1/ml$, where $\hat{f}_1$ is the optimal value of the cluster function f_1, m is the number of points in the set A and l is the number of clusters. If $d_1(\bar{x}, \tilde{x}) \leq \varepsilon$ for two points $\bar{x}, \tilde{x} \in \bar{A}_4$, then we remove one of them and keep another one (with the lowest value of the auxiliary cluster function).

Reduction using the triangle inequality. The distance function d_1 satisfies the triangle inequality. We use this statement to reduce the number of distance function calculations both in the k-medians and in the partial k-medians algorithms.

First, we consider Algorithm 1. Assume that $x^1, \ldots, x^{l-1}$ is the solution to the $(l-1)$-partition problem. Let $\hat{x}$ be a current approximation to the solution of the auxiliary clustering problem. Compute $d_1(\hat{x}, x^j), j \in J_{l-1}$, and assume that $a \in A^j$ for some $j \in J_{l-1}$. According to the triangle inequality we have

$$d_1(\hat{x}, x^j) \leq d_1(a, \hat{x}) + d_1(a, x^j) = d_1(a, \hat{x}) + r_{l-1}^a,$$

or $\quad d_1(a, \hat{x}) \geq d_1(\hat{x}, x^j) - r_{l-1}^a,$

where r_{l-1}^a is defined in (4). This means that if $d_1(\hat{x}, x^j) > 2r_{l-1}^a$, then $d_1(a, \hat{x}) > r_{l-1}^a$. Therefore, there is no need to calculate $d_1(a, \hat{x})$ as the point a does not belong to the cluster with the center $\hat{x}$. This scheme allows us to significantly reduce the number of distance function evaluations as the number of clusters increases.

Now consider the k-medians algorithm. Let $\hat{x}^1, \ldots, \hat{x}^l$ be a current approximation to the solution of the l-partition problem. Compute $d_1(\hat{x}^q, \hat{x}^j)$ for

$q, j \in J_l, q \neq j$. Assume that for a given point $a \in A$ distances $d_1(a, \hat{x}^q)$, $q = 1, \dots, j$ have been calculated for some $j \in J_{l-1}$. Let $\tilde{x} \in \{\hat{x}^1, \dots, \hat{x}^j\}$ be such that $d_1(a, \hat{x}) = \min_{q=1,\dots,j} d_1(a, \hat{x}^q)$. Applying the similar approach as in the case of Algorithm 1 we get that if $d_1(\tilde{x}, \hat{x}^{j+1}) > 2d_1(a, \tilde{x})$, then there is no need to calculate $d_1(a, \hat{x}^{j+1})$ as the point a does not belong to the cluster A^{j+1} with the center $\hat{x}^{j+1}$.

Table 1. Brief description of data sets.

Data sets	m	n	n_c
Page Blocks	5473	10	5
Gas Sensor Array Drift	13910	128	6
EEG Eye State	14980	14	2
Letter Recognition	20000	16	26
KEGG Metabolic Relation Network	53413	20	–
Sensorless Drive Diagnosis	58509	48	11
Pla85900	85900	2	–
Localization Data for Person Activity	164860	3	11

5. Numerical experiments

We carried out numerical experiments using 8 real-world data sets to evaluate the performance of the AkMed algorithm and to compare it with some other clustering algorithms. Results of these experiments are discussed in this section.

Data sets. The brief description of data sets is given in Table 1, and their detailed descriptions can be found in (Dua & Graff, 2019; Reinelt, 1991). These data have only numeric attributes and no missing values. Some of them have class labels (we denote by n_c number of classes and use "–" if a data set has no class label). We computed up to 25 clusters in all data.

Performance measures. The performance measures used in our evaluations are:

- *Davies-Bouldin cluster validity index* (Davies & Bouldin, 1979): to determine how well-separated and compact the clusters are;
- *silhouettes* (Rousseeuw, 1987): to demonstrate how well a data point is clustered;
- *purity* (Dhillon, Fan, & Guan, 2001): to show how well the cluster distribution reflects the existing class structure of the data. It is applicable when the class information is available.

In addition, in our comparisons, we use the relative error E_{AL} computed as $E_{AL} = \frac{\tilde{f} - f_{best}}{f_{best}} \times 100\%$, where f_{best} (multiplied by the number shown after the name of each data set) is the best known value of the cluster function (2) (multiplied by m) for the corresponding number of clusters, and $\tilde{f}$ is the value of the clustering function (2) obtained by an algorithm AL.

Clustering algorithms for comparison. We use the following clustering algorithms:

- multi-start *k-medians* (MkMed) algorithm;
- global *k-medians* (GkMed) algorithm;
- *k-medians++* (kMed++) algorithm;
- *smooth clustering* (SC) algorithm introduced in (Bagirov & Mohebi, 2016);
- incremental DC clustering (IDCC) algorithm proposed in (Bagirov & Taheri, 2017).

The GkMed algorithm is a special case of the AkMed algorithm where the data point providing the largest decrease of the cluster function is selected as a starting point for the next cluster center. The SC and IDCC algorithms use different approaches to the one utilized in this paper. In the paper (Bagirov & Mohebi, 2016), the objective functions in both the clustering and the auxiliary clustering problems are approximated by smooth functions, and the problems are solved by applying a sequential clustering algorithm. In (Bagirov & Taheri, 2017), the objective functions are formulated using their DC representations. Then the DC functions are smoothed partially and a DC algorithm is applied to solve the clustering and the auxiliary clustering problems with the partially smoothed functions.

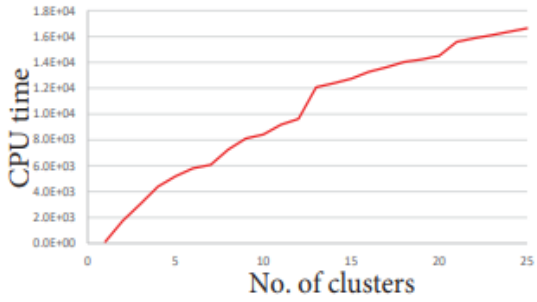
Implementation of algorithms. All algorithms except kMed++ were implemented in Fortran 95 and compiled using the *gfortran* compiler. We utilize MATLAB code of kMed++ available in (Burkardt, 2021). Computational results are obtained on a PC with the Intel(R) Core(TM) i5-2400S CPU 2.50 GHz and RAM 8 GB.

The parameters γ_1, γ_2 and γ_3 in the AkMed algorithm are selected according to the recommendations given in (Ordin & Bagirov, 2015). The details of implementation and parameter selection of the SC and the IDCC algorithms can be found in (Bagirov & Mohebi, 2016) and (Bagirov & Taheri, 2017), respectively. For the MkMed algorithm, the maximum number of starting points was kept big enough and its running time was allowed to be no less than twice of the CPU time used by the AkMed algorithm. The CPU time used by all algorithms was limited to 20 hours.

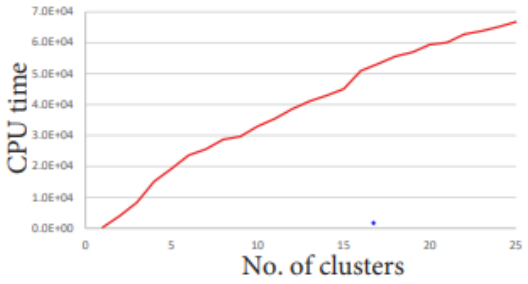
5.1. *Performance of the AkMed algorithm*

To demonstrate the performance of the AkMed algorithm using evaluation measures mentioned above, we select four representative data sets from Table 1.

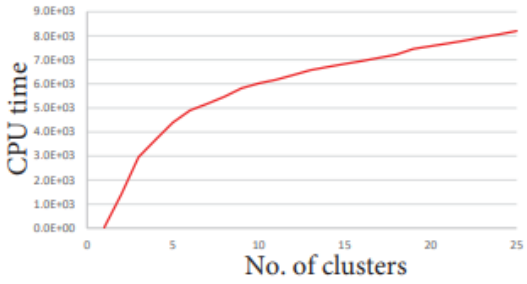
CPU time. Figure 4 illustrates the dependence of the computational time required by the AkMed algorithm on the number of clusters. It can be observed from these figures that there is no any significant increase of the computational time as the number of clusters increases.



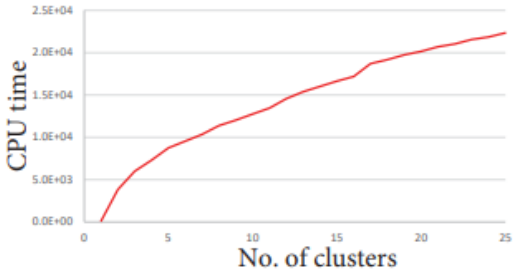
(a) KEGG Metabolic Relation Network



(b) Sensorless Drive Diagnosis



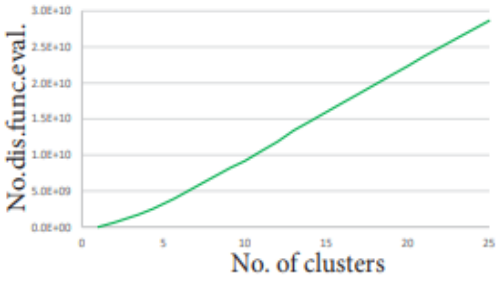
(c) Pla85900



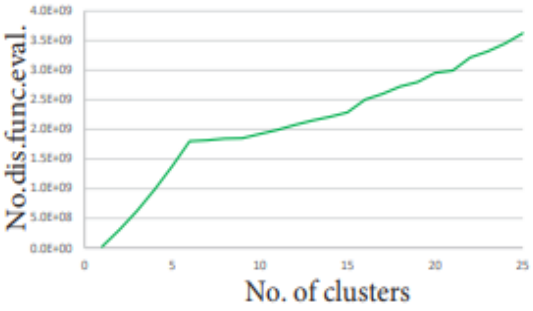
(d) Localization Data for Person Activity

Fig. 4. CPU time vs. number of clusters

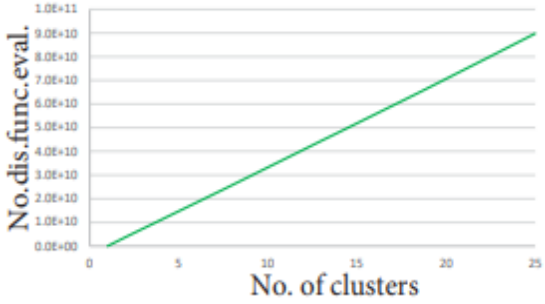
Number of distance function evaluations. The dependence of the number of distance function evaluations on the number of clusters is given in Figure 5. The results show that increasing the number of clusters does not lead to a significant increase of the number of distance function evaluations.



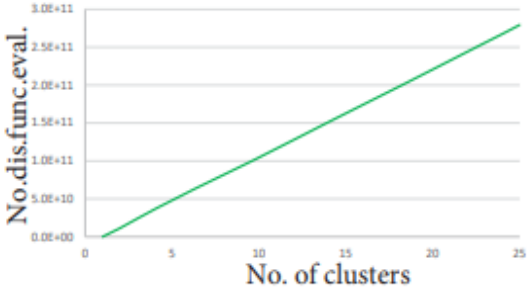
(a) KEGG Metabolic Relation Network



(b) Sensorless Drive Diagnosis



(c) Pla85900



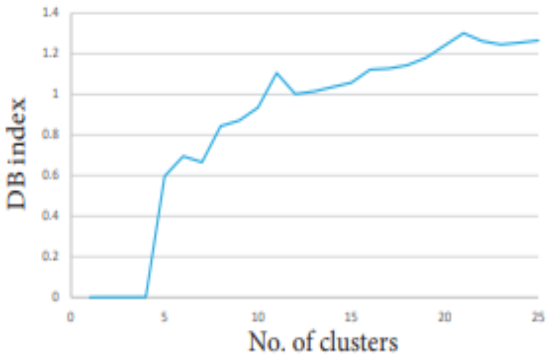
(d) Localization Data for Person Activity

Fig. 5. Number of distance function evaluations vs. number of clusters.

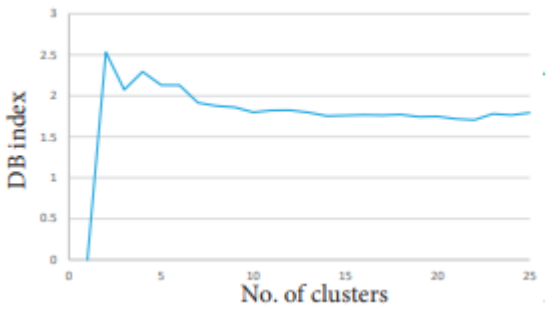
Davies-Bouldin cluster validity index. The DB indices obtained by the AkMed algorithm are given in Figure 6. Since all data sets contain a large number of clusters, we consider the DB indices only for $k > 5$. According to the DB index the number of compact and well-separated clusters obtained by the AkMed algorithm are as follows: Page Blocks - 9; EEG Eye State - 7; Letter Recognition - 22; Sensorless Drive Diagnosis - 13.

Silhouette plots. We draw the silhouette plots for the AkMed algorithm using data set: Page Blocks with $k = 2, 3, 5, 10$. The plots for this data set are displayed in Figure 7. In all cases there is one large cluster whose points are well seated in this cluster. Other clusters contain misclassified instances, however, the number of misclassified points tends to decrease as the number of clusters increases. Note that Page Blocks data set has one very large class containing almost 89% of all instances. In addition, this data set also contains noise. These plots show that the AkMed algorithm is capable of finding well-separated clusters even in such data sets.

(a) Page Blocks



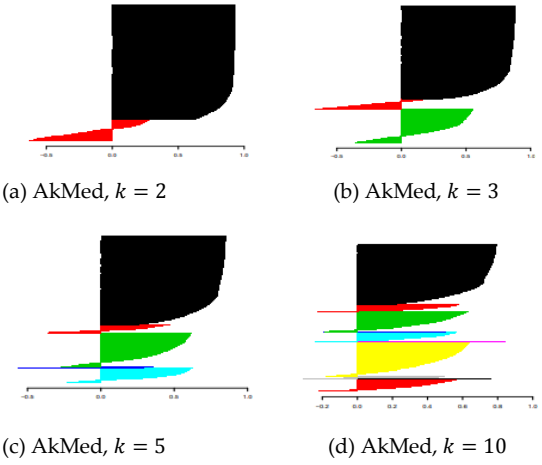
(b) EEG Eye State



(c) Letter Recognition

(d) Sensorless Drive Diagnosis

Fig. 6. Results for DB index.

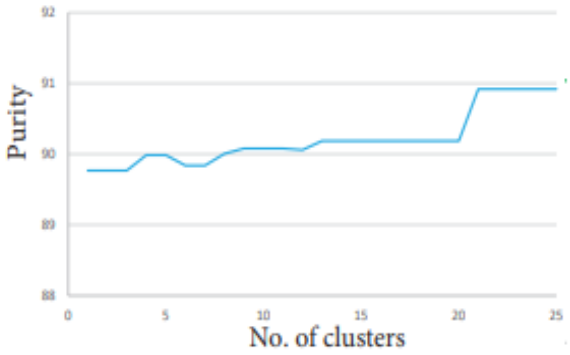


(a) AkMed, $k = 2$ (b) AkMed, $k = 3$

(c) AkMed, $k = 5$ (d) AkMed, $k = 10$

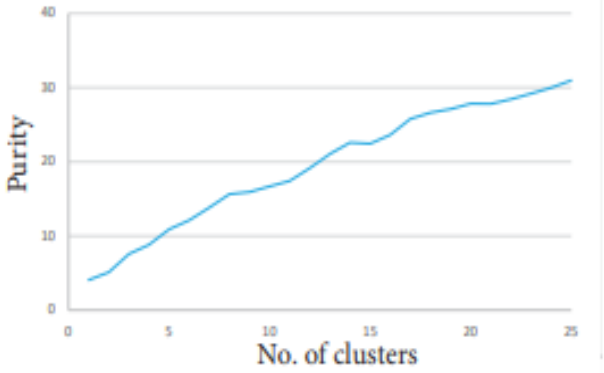
Fig. 7. Silhouette plots for Page Blocks data set.

Purity. Results for the purity of clusters obtained by the AkMed algorithm are depicted in Figure 8. The purity depends on the number of clusters and existing classes. In general, the purity increases as the number of clusters increases. However, this observation is not always true for EEG Eye State data set. This is due to the fact in this data set one point is far away from the rest of the data set. For the 2-clustering problem one of the clusters contain only this point.

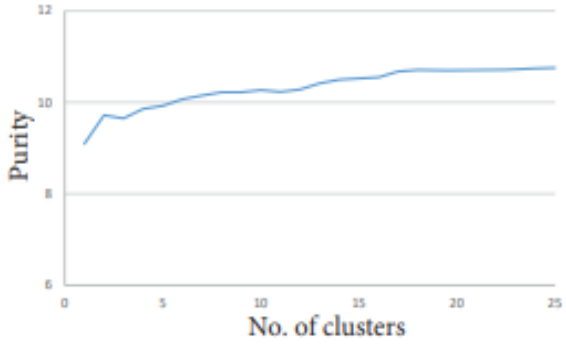


(a) Page Blocks

(b) EEG Eye State



(c) Letter Recognition



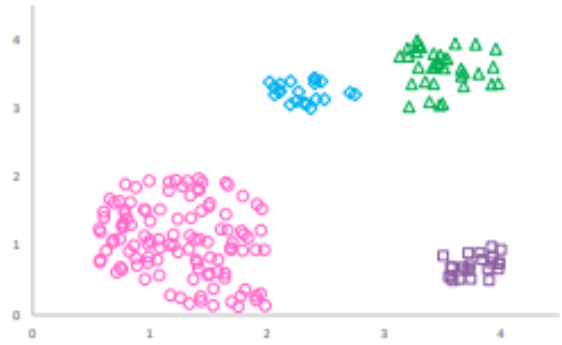
(d) Sensorless Drive Diagnosis

Fig. 8. Results for Purity.

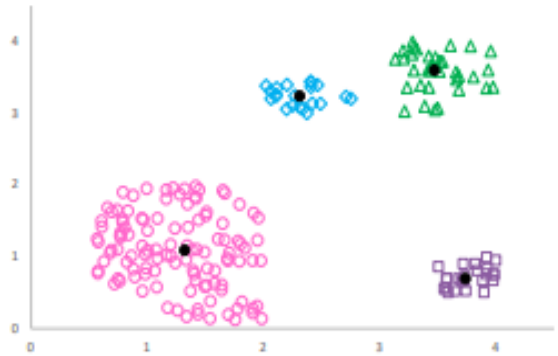
Results on the robustness. To demonstrate the robustness of the AkMed algorithm to outliers we consider a synthetic data set in the 2-dimensional space. This data set is given in Figure 9 where clustering results in data sets both with outliers and without outliers are depicted. There are four clusters in this data and cluster centers are shown using black dots. We can see that adding outliers does not lead to any changes of cluster structure. Moreover, there is no significant changes in locations of cluster centers. We can conclude that the AkMed algorithm is robust to outliers in this data set.

5.2. *Comparison with other clustering algorithms*

Next, we compare the AkMed algorithm with other clustering algorithms mentioned above. We use optimal values of the cluster function obtained by algorithms to compare their accuracy. The results are given in Table 2. The notation "fail" used in this table means that an algorithm fails to compute clusters in the given time frame.

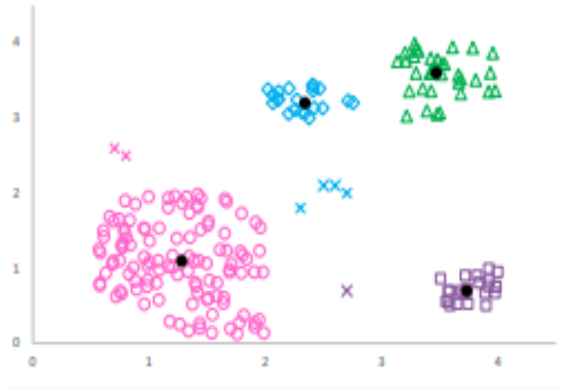


(a) Data set with no outliers



(b) Clustering results with no outliers

(c) Data set with outliers



(d) Clustering results with outliers

Fig. 9. Robustness of AkMed algorithm.

We can see from Table 2 that the AkMed algorithm outperforms (in some cases substantially) the MkMed algorithm in the data sets: EEG Eye State ($k \geq 15$), KEGG Network ($k \geq 10$) and Localization Data for Person Activity. The MkMed algorithm fails to find any solution (in the given time frame) in Localization Data for Person Activity data set whereas the AkMed algorithm is able to find solutions with high accuracy. In other data sets the performances of these two algorithms are similar. We can also see that in most data sets the AkMed and MkMed algorithms perform similarly when the number of clusters k is small.

Overall, the AkMed algorithm outperforms the GkMed algorithm. However, the difference is not significant as the GkMed algorithm is a special case of the AkMed algorithm. Results for Page Blocks data set show that the AkMed algorithm outperforms the GkMed algorithm when a data set does not have well separated clusters. The AkMed algorithm is more accurate than the kMed++ algorithm in all data sets.

Furthermore, the results show that the AkMed algorithm outperforms the SC algorithm in the data sets: Page Blocks, Letter Recognition, KEGG Network and Pla85900. In other data sets the performances of these algorithms are similar. The SC algorithm fails to find solution in Gas Sensor Array Drift ($k \geq 10$). This data set contains a large number of attributes and results demonstrate that the SC algorithm becomes highly time consuming in such data sets.

Finally, the AkMed algorithm has a better performance than the IDCC algorithm in the data sets: Page Blocks ($k \geq 10$), Gas Sensor Array Drift ($k \geq 5$), KEGG Network ($k \geq 20$) and Sensorless Drive Diagnosis ($k \geq 20$). In other data sets these algorithms perform similarly. Results presented in this subsection demonstrate that in these eight data sets the performance of the AkMed algorithm in finding accurate solutions is either similar or better than that of other clustering algorithms used for comparison.

Table 2: Relative errors for different clustering algorithms

k	f_{best}	E_{AkMed}	E_{MkMed}	E_{GkMed}	$E_{\text{kMed++}}$	E_{SC}	E_{IDCC}	f_{best}	E_{AkMed}	E_{MkMed}	E_{GkMed}	$E_{\text{kMed++}}$	E_{SC}	E_{IDCC}
	Page Blocks (×E6)							**Gas Sensor Array Drift (×E9)**						
2	8.41	0.00	25.28	0.00	68.85	0.00	0.00	2.27	0.00	0.00	0.00	3.48	0.00	0.95
3	6.75	0.00	0.00	0.00	49.10	0.00	0.00	1.90	0.00	0.00	0.00	2.65	0.00	0.88
5	4.88	0.00	0.00	0.00	27.06	0.00	0.03	1.45	0.00	0.00	0.00	2.22	0.01	8.37
10	3.07	0.00	17.28	0.01	14.77	3.36	3.52	1.06	0.00	0.92	0.02	1.57	fail	5.73
15	2.48	0.00	27.04	1.78	10.03	3.33	4.64	0.89	0.45	0.00	0.50	1.39	fail	7.50
20	2.18	0.00	26.06	0.44	7.95	0.61	3.05	0.78	0.00	1.83	0.00	1.26	fail	8.09
25	1.95	0.00	30.40	0.62	6.85	1.58	4.13	0.71	0.00	2.87	0.00	1.12	fail	8.36
	EEG Eye State (×E6)							**Letter Recognition (×E5)**						
2	5.29	0.00	15.46	0.00	14.83	0.00	0.00	4.83	0.00	0.00	0.00	1.86	0.02	0.00
3	4.20	0.00	38.65	0.00	4.37	0.00	0.00	4.58	0.00	0.07	0.00	1.73	5.67	0.03
5	2.94	0.00	86.96	0.00	0.06	0.00	0.00	4.23	0.00	0.04	0.00	1.55	8.79	1.65
10	2.17	0.00	139.40	0.00	0.02	0.04	0.04	3.76	0.00	0.57	0.59	1.29	9.07	0.35
15	1.97	0.00	156.50	0.00	0.17	0.00	0.36	3.52	0.00	0.30	0.02	1.13	3.85	0.05
20	1.83	0.21	170.84	0.21	0.15	0.00	0.03	3.33	0.06	0.30	0.00	1.03	4.47	0.02
25	1.74	0.00	140.01	0.18	0.14	0.08	0.20	3.19	0.02	0.62	0.00	0.98	3.40	0.05

	KEGG Network (×E6)							Sensorless Drive Diagnosis (×E6)						
2	3.59	0.00	0.00	0.00	3.56	1.08	0.12	1.25	0.00	0.00	0.00	0.30	0.00	0.01
3	2.80	0.00	0.00	0.00	3.71	0.04	0.27	1.17	0.00	0.00	0.00	0.24	0.13	0.23
5	2.08	1.19	1.50	1.19	8.55	1.26	0.00	1.08	0.00	0.26	0.00	0.18	0.52	0.16
10	1.44	0.00	4.55	0.02	39.74	4.64	0.91	0.94	0.00	0.00	0.06	1.10	0.34	0.24
15	1.20	0.00	7.27	0.00	28.58	0.98	1.12	0.87	0.00	0.51	0.01	8.21	0.85	0.15
20	1.06	0.00	15.08	0.80	26.65	1.68	1.62	0.82	0.00	1.05	0.00	53.23	0.54	6.17
25	0.97	0.00	16.60	0.00	19.82	2.79	2.14	0.79	0.00	0.63	0.02	45.34	0.73	10.35
	Pla85900 (×E10)							Localization Data for Person Activity (×E5)						
2	2.07	0.00	0.42	0.38	1.84	0.19	0.00	1.76	0.00	fail	0.00	4.88	0.00	0.00
3	1.63	0.00	0.29	0.00	1.41	0.51	0.27	1.52	0.17	fail	0.17	4.10	0.00	0.00
5	1.26	0.15	0.00	0.15	1.07	1.80	0.12	1.27	0.00	fail	0.00	3.41	0.00	0.00
10	0.89	0.08	0.07	0.10	7.67	1.57	0.00	0.96	0.01	fail	0.01	2.48	0.00	0.01
15	0.73	0.23	0.00	0.23	6.34	1.31	0.07	0.85	0.01	fail	0.01	2.11	0.00	0.01
20	0.64	0.11	0.00	0.41	5.57	1.24	0.22	0.77	0.00	fail	0.00	fail	0.07	0.00
25	0.57	0.10	0.05	0.10	5.02	1.62	0.00	0.72	0.00	fail	0.00	fail	0.18	0.00

6. Conclusions and discussions

In this paper, we introduced the adaptive k-medians (AkMed) clustering algorithm where the similarity measure is defined using the L_1-norm. The AkMed algorithm computes clusters gradually starting from one cluster which is the median of the whole data set and adds one cluster center at each iteration. An auxiliary clustering problem is used to design an algorithm for finding initial cluster centers. Both the clustering and the auxiliary clustering problems are solved using the k-medians algorithm. We presented the estimation of the number of distance function evaluations required by the AkMed algorithm and described different approaches to reduce this number.

The proposed algorithm was tested and compared with other clustering algorithms using 8 real-world data sets. Selected data sets are diverse in the sense of the number of data points and the number of attributes. Based on the results of the numerical experiments we draw the following conclusions:

- CPU time and the number of distance function calls required by the AkMed algorithm increase linearly or almost linearly as the number of clusters increases;
- the AkMed algorithm is robust to outliers when they are well separated from other data points and constitute a small portion of a data set. This claim is not conclusive as more detailed research is required to study the robustness of the AkMed algorithm depending on different types of outliers;
- the AkMed algorithm is able to find accurate solutions in more data sets than any other clustering algorithms used in numerical experiments;
- using the Davies-Bouldin cluster validity index we can see that the AkMed algorithm is able to compute the small number of well-separated and compact clusters. Moreover, using the silhouette plots we showed that most data points are well located in clusters found by the AkMed algorithm;
- the AkMed algorithm has some limitations. This algorithm becomes time consuming in data sets containing hundreds of thousands of data points and therefore, it is not applicable to very large data sets. Although the AkMed algorithm is able to find compact clusters, results on silhouettes show that these clusters are not always well separated. These drawbacks will be addressed in future research.

References

Aggarwal C, Hinneburg A, Keim D. (2001). On the surprising behavior of distance metrics in high dimensional space. In: ICDT '01 Proceedings of the 8th International Conference on Database Theory (pp.420–434).

Bagirov A, Karmitsa N, Taheri S. (2020). Partitional Clustering via Nonsmooth Optimization. Springer, Cham..

Bagirov A, Mohebi E. (2016). An algorithm for clustering using L1-norm based on hyperbolic smoothing technique. Computational Intelligence, 32, 439–57.

Bagirov A, Ordin B, Mohebi E. (2020). An incremental nonsmooth optimization algorithm for clustering using L1 and l∞ norms. Journal of Industrial and Management Optimization, 16(6), 2757-2779.

Bagirov A, Taheri S. (2017). A DC optimization algorithm for clustering problems with L1-norm. Iranian Journal of Operations Research, 8(2), 2–24.

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

Bagirov A. (2008). Modified global k-means algorithm for minimum sum-of-squares clustering problems. Pattern Recognition 41(10), 3192–3199.

Bai L, Liang J, Sui C, Dang C. (2013). Fast global k-means clustering based on local geometrical information. Information Sciences, 245, 168–80.

Ben-David S. (2020). Computational feasibility of clustering under clusterability assumptions. 2020. https://arxiv.org/abs/1501.00437.

Burkardt J. (2021) https://people.sc.fsu.edu/~jburkardt/

Carmichael J, Sneath P. (1969). Taxometric maps. Systematic Zoology, 18, 402–415.

Castellanos A, Cigarran J, Garcıa-Serrano A. (2017). Formal concept analysis for topic detection: A clustering quality experimental analysis. Information Systems, 66, 24-42.

Dai Q, Xiong Z, Xie J, Wang X, Zhang Y, Shang J. (2019). A novel clustering algorithm based on the natural reverse nearest neighbor structure. Information Systems, 84, 1-16.

Davies D, Bouldin D. (1979). A cluster separation measure. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1(2), 224-227.

Dhillon, I.S., Fan, J., Guan, Y. (2001). Efficient clustering of very large document collections. In: Grossman, R.L., Kamath, C., Kegelmeyer, P., Kumar, V., Namburu, R.R. (eds) Data Mining for Scientific and Engineering Applications. Massive Computing, vol.2., pp.357-381.

Dua D, Graff C. (2019). UCI Machine Learning Repository. University of California, Irvine, School of Information and Computer Sciences, http: //archive.ics.uci.edu/ml.

Friggstad Z, Khodamoradi K, Rezapour M, Salavatipour M. (2019). Approximation schemes for clustering with outliers. ACM Transactions on Algorithms,15(2), 1-26.

Hanilci C, Ertas F. (2011). Comparison of the impact of some Minkowski metrics on vq/gmm based speaker recognition. Computers and Electrical Engineering, 37, 41-56.

Jain A. (2010). Data clustering: 50 years beyond k-means. Pattern Recognition Letters, 31(8), 651-666.

Jajuga K. (1987). A clustering method based on the L1-norm. Computational Statistics & Data Analysis, 5(4), 357-371.

Karmitsa N, Bagirov A, Taheri S. (2017). New diagonal bundle method for clustering problems in large data sets. European Journal of Operational Research, 263(2), 367-379.

Karmitsa N, Bagirov A, Taheri S. (2018). Clustering in large data sets with the limited memory bundle method. Pattern Recognition, 83, 245–259.

Khachay M, Khachay D. (2019). Attainable accuracy guarantee for the k-medians clustering in [0,1]. Optimization Letters, 13, 1837-1853.

Kumar A, Sabharwal Y, Sen S. (2010). Linear-time approximation schemes for clustering problems in any dimensions. Journal of the ACM, 57(2), Article 5.

Lai J, Huang TJ. (2010). Fast global k-means clustering using cluster membership and inequality. Pattern Recognition, 43(5), 1954-1963.

Likas A, Vlassis N, Verbeek J. (2003). The global k-means clustering algorithm. Pattern Recognition, 36(2), 451-461.

Ordin B, Bagirov A. (2015). A heuristic algorithm for solving the minimum sum-ofsquares clustering problems. Journal of Global Optimization, 61(2), 341-361.

Reinelt G. (1991). TSP-LIB-A Traveling Salesman Library. ORSA Journal on Computing, 3, 319-350.

Rousseeuw P. (1987). Silhouettes: Graphical aid to interpret and validate of cluster analysis. Journal of Computational and Applied Mathematics, 20, 53-65.

Sabo K, Scitovski R, Vazler I. (2013). One-dimensional center-based L1-clustering method. Optimization Letters, 7(1), 5-22.

de Souza R, de Carvalho F. (2004). Clustering of interval data based on city-block distances. Pattern Recognition Letters, 25, 353-365.

Spath H. (1976). Algorithm 30: L_1 cluster analysis. Computing, 16(4), 379-387.

Xavier A, Xavier V. (2011). Solving the minimum sum-of-squares clustering problem by hyperbolic smoothing and partition into boundary and gravitational regions. Pattern Recognition, 44(1), 70-77.

Zhang J, Peng L, Zhao X, Kuruoglu E. (2012). Robust data clustering by learning multimetric Lq-norm distances. Expert Systems with Applications, 39, 335-349.