JOURNAL OF INDUSTRIAL AND
MANAGEMENT OPTIMIZATION
Volume **16**, Number **6**, November **2020**

doi:10.3934/jimo.2019079

pp. **2757–2779**

AN INCREMENTAL NONSMOOTH OPTIMIZATION ALGORITHM FOR CLUSTERING USING L_1 AND L_∞ NORMS

Burak Ordin*

Department of Mathematics, Faculty of Science
Ege University, Bornova, 35100, Izmir, Turkey

Adil Bagirov

School of Mathematical Sciences, Chongqing Normal University, Chongqing, China and
School of Science, Engineering and Information Technology, Federation University Australia
Ballarat, Victoria, 3353, Australia

Ehsan Mohebi

Federation University Australia, Ballarat, Victoria, 3353, Australia

(Communicated by Jan Joachim Ruckmann)

Abstract. An algorithm is developed for solving clustering problems with the similarity measure defined using the L_1 and L_∞ norms. It is based on an incremental approach and applies nonsmooth optimization methods to find cluster centers. Computational results on 12 data sets are reported and the proposed algorithm is compared with the X-means algorithm.

1. Introduction.

The clustering deals with the problem of organization of a collection of patterns into clusters based on similarity. It has found many applications in medicine, bioinformatics, engineering, business, information retrieval [25, 30, 44].

Most clustering algorithms are based on the hierarchical and partitional approaches. Algorithms based on the hierarchical approach generate a dendrogram representing the nested grouping of patterns and similarity levels at which groupings change [26]. Partitional clustering algorithms find the partition that optimizes a clustering criterion [26].

A measure of similarity between two patterns is fundamental to the definition of a cluster [26]. This measure, in particular, is defined using various distance functions. The use of different similarity measures may help to better understand the intrinsic clustering structure of a data set. The widely used similarity measure is the squared Euclidean distance. In this case the clustering is known as the minimum sum-of-squares clustering problem. Various algorithms have been developed to solve it (see, for example, [24, 25, 44]).

Clustering problems with the similarity measures defined using other Minkowski norms, including the L_1 and L_∞ norms, have attracted significantly less attention than the minimum sum-of-squares clustering problem. In this paper, an algorithm is developed to solve clustering problems with the similarity measure defined using the

2020 *Mathematics Subject Classification.* Primary: 90C90; Secondary: 90A05.

Key words and phrases. Nonsmooth optimization, nonconvex optimization, cluster analysis, unsupervised learning, incremental algorithm.

* Corresponding author: Burak Ordin.

L_1 and L_∞ norms. This algorithm computes clusters gradually starting from one cluster which is the whole data set. An auxiliary clustering problem is introduced to find starting cluster centers. Using computational results it is demonstrated that the algorithm is efficient for solving clustering problems in large data sets.

The rest of the paper is organized as follows. Section 2 provides a brief review of the related work. The nonsmooth optimization formulation of the clustering problem is given in Section 3. Algorithms for solving clustering problems are discussed in Section 4. Section 5 presents an algorithm for finding starting cluster centers. An incremental clustering algorithm and its implementation are discussed in Section 6. Computational results are reported in Section 7. Section 8 concludes the paper.

2. Related work.

In this section we give a brief review of algorithms for solving clustering problems with the similarity measures defined by the L_1 and L_∞ norms. Let $d : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}_+$ be a distance function. Here $\mathbb{R}_+ = \{u \in \mathbb{R} : u \geq 0\}$. For $x, y \in \mathbb{R}^n$ this function can be defined using the L_p-norm:

$$d(x,y) \equiv d_p(x,y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{1/p}, \ p \geq 1.$$

In particular, the distance function using the L_1-norm is defined as:

$$d_1(x,y) = \sum_{i=1}^{n} |x_i - y_i|,$$

and using the L_∞-norm it is:

$$d_\infty(x,y) = \max_{i=1,\dots,n} |x_i - y_i|.$$

Note that both distance functions are nonsmooth. We will also consider the distance-like function d_2^2 which is based on the squared Euclidean norm:

$$d_2^2(x,y) = \sum_{i=1}^{n} (x_i - y_i)^2.$$

In this paper, we use the functions d_1 and d_∞ to define similarity measures. Clustering results with such functions can be easily interpreted.

There are many applications where clustering algorithms with the functions d_1 and d_∞ obtain significantly better results than those with the function d_2^2. In high dimensional data mining applications the function d_1 is consistently more preferable than the function d_2^2 [1]. Clustering algorithms with d_1 and d_∞ are more robust to the so-called severe outliers than those with d_2^2 [46]. The functions d_1 and d_∞ achieve the highest identification and best verification rates in closed-set text-independent speaker recognition systems [23].

The paper [18] (see, also [29]) seems to be the first paper, where the distance function d_1 was used to define the similarity measure in clustering problems. The paper [40] presents an algorithm for solving clustering problems with the function d_1, where cluster centers are computed as medians.

The ISODATA clustering algorithm [41] using the function d_1 was introduced in [27], where the comparison with the distance function d_2^2 demonstrates that the algorithm with d_1 is superior when the correlation coefficient is high and negative. In [21], the authors examine the use of a range of Minkowski norms for the clustering. The X-means algorithm introduced in [33] can use various distance functions as similarity measures. In the paper [39], the authors introduce adaptive and non-adaptive clustering methods using the function d_1. A one dimensional center-based L_1-clustering algorithm is proposed in [37].

An algorithm, called the Hyperbox clustering with Ant Colony Optimization, is proposed in [35] to solve problems with the function d_∞. The paper [22] presents an approach to the fuzzy clustering using the L_1-norm. In the paper [45], the authors give a fuzzy c-means algorithm with the L_1-norm and fuzzy covariance. The c-mean algorithms with the d_1 and d_∞ functions are introduced in [16]. The hyperbolic smoothing technique is applied to solve clustering problems with the L_1-norm in [9]. The comparison of different algorithms for clustering with the L_1 and L_∞-norms is presented in [8].

Most algorithms mentioned above can efficiently solve clustering problems only in small and medium sized data sets. They can only find local solutions and such solutions might be significantly different from global solutions in large data sets. Furthermore, these algorithms are sensitive to the choice of starting points. The clustering is a global optimization problem, it has many solutions and only global or near global solutions provide the best cluster structure of a data set. Conventional global optimization algorithms are not efficient for solving such problems in large data sets. In this paper, we develop an algorithm which can find global or near global solutions to clustering problems with the d_1 and d_∞ distance functions.

3. **Clustering and auxiliary clustering problems.**

In this section we give the nonsmooth optimization formulation of the clustering and auxiliary clustering problems.

3.1. **Nonsmooth nonconvex formulation of clustering problems.**

Assume that a finite set A of points in the n-dimensional space $\mathbb{R}^n$ is given, that is $A = \{a^1, \dots, a^m\}$, where $a^i \in \mathbb{R}^n,\ i = 1, \dots, m$. The hard unconstrained partitional clustering problem is the distribution of the points of the set A into a given number k of disjoint subsets $A^j,\ j = 1, \dots, k$ such that $A^j \neq \emptyset,\ j = 1, \dots, k$ and:

1) $A^j \bigcap A^l = \emptyset,\ j, l = 1, \dots, k,\ j \neq l$;
2) $A = \bigcup_{j=1}^{k} A^j$;
3) no constraints are imposed on the clusters $A^j,\ j = 1, \dots, k$.

The sets $A^j,\ j = 1, \dots, k$ are called clusters. Each cluster A^j can be identified by its center $x^j \in \mathbb{R}^n,\ j = 1, \dots, k$. The nonsmooth nonconvex optimization formulation of the clustering problem is as follows [11, 12, 15, 17]:

$$\text{minimize } f_k(x) \text{ subject to } x = (x^1, \dots, x^k) \in \mathbb{R}^{kn}, \tag{1}$$

where

$$f_k(x^1, \dots, x^k) = \frac{1}{m} \sum_{i=1}^{m} \min_{j=1,\dots,k} d(x^j, a^i). \tag{2}$$

We call f_k *the k-th cluster function* and the problem (1) *the k-clustering problem.* It is obvious that if the distance-like function $d(x, a)$ is convex with respect to x then the function f_1 is convex. This is true when one uses d_1, d_2^2 and d_∞ functions. The function f_k is nonconvex and nonsmooth for $k \geq 2$ due to the minimum operation in its definition. It is nonsmooth for any $k \geq 1$ when the distance functions d_1 and d_∞ are applied.

3.2. **An auxiliary clustering problem.** Given the solution $x^1, \dots, x^{l-1},\ l > 1$ to the $(l-1)$-clustering problem define

$$r^i_{l-1} = \min_{j=1,\dots,l-1} d(x^j, a^i),$$

which is the distance between the data point $a^i,\ i = 1, \dots, m$ and its cluster center. We will also use the notation r^a_{l-1} for the data point $a \in A$. The function:

$$\bar{f}_l(y) = \frac{1}{m} \sum_{i=1}^{m} \min\left\{r^i_{l-1}, d(y, a^i)\right\},\ y \in \mathbb{R}^n \tag{3}$$

is called *the l-th auxiliary cluster function* [4,10,32]. It is nonsmooth and nonconvex. Let f^*_{l-1} be the optimal value of the problem (1) for $k = l-1$. Then

$$f^*_{l-1} = \frac{1}{m} \sum_{i=1}^{m} r^i_{l-1}.$$

It is obvious that

$$\bar{f}_l(y) \le f^*_{l-1}\ \forall y \in \mathbb{R}^n. \tag{4}$$

The l-th auxiliary clustering problem is formulated as follows:

$$\text{minimize } \bar{f}_l(y) \text{ subject to } y \in \mathbb{R}^n. \tag{5}$$

4. Solving optimization problems.

There are three different optimization problems to be solved to find cluster centers: (i) the problem of finding a center of one cluster; (ii) the problem (1) to find all cluster centers; and (iii) the problem (5) to find starting points for cluster centers. In this section we discuss algorithms for solving each of these problems.

4.1. **Finding a center of one cluster.** A center x for a cluster B is found as a solution to the following optimization problem:

$$\text{minimize } \varphi(x) = \frac{1}{|B|} \sum_{b \in B} d(x, b) \text{ subject to } x \in \mathbb{R}^n. \tag{6}$$

Here $|B|$ stands for the cardinality of the set B.

If the function d is defined using the squared Euclidean norm then the centroid of the cluster B is the solution to the problem (6) and the centroid can be easily computed without solving the optimization problem.

If the function d is defined using the L_1-norm then the median of the cluster B is the solution to the problem (6). This result was first used in the k-median algorithm introduced in [40]. In this paper we use the nonsmooth optimization techniques to prove it.

Proposition 1. *If in (6) the function d is the distance function d_1 then the median of the set B is the solution to Problem (6).*

Proof. For the distance function d_1 the function φ can be written as follows:

$$\varphi(x) = \frac{1}{|B|} \sum_{b \in B} \sum_{i=1}^{n} |x_i - b_i| = \frac{1}{|B|} \sum_{i=1}^{n} \sum_{b \in B} |x_i - b_i|.$$

Consider functions

$$\psi_i(x_i) = \sum_{b \in B} |x_i - b_i| = \sum_{b \in B} \max\{x_i - b_i, b_i - x_i\},\ i = 1, \dots, n.$$

The function φ is separable and can be represented as:

$$\varphi(x) = \frac{1}{|B|} \sum_{i=1}^{n} \psi_i(x_i).$$

This means that the minimization of φ is equivalent to the minimization of the functions $\psi_i, i \in \{1, \dots, n\}$. We assume that for any $i \in \{1, \dots, n\}$ coordinates b_i are different for all $b \in B$ (if there are any two $b^1, b^2 \in B$ such that $b_i^1 = b_i^2$ one of them can be removed). For $i \in \{1, \dots, n\}$ consider the following sets:

$$B^- = \{b \in B : b_i < x_i\},\ B^+ = \{b \in B : b_i > x_i\},\ B^0 = \{b \in B : b_i = x_i\}.$$

Since all numbers b_i ($b \in B$) are different it is obvious that for a given $x \in \mathbb{R}^n$ the cardinality $|B^0|$ of the set B^0 is either 0 or 1. Then the subdifferential of the function ψ_i at x_i is:

$$\begin{aligned} \partial\psi_i(x_i) &= |B^-| - |B^+| + \left[-|B^0|, |B^0|\right] \\ &= \left[|B^-| - |B^+| - |B^0|, |B^-| - |B^+| + |B^0|\right]. \end{aligned}$$

The point x_i to be a global minimizer of the function ψ_i it is necessary and sufficient that $0 \in \partial\psi_i(x_i)$. This means that at x_i

$$|B^-| - |B^+| - |B^0| \le 0,\ |B^-| - |B^+| + |B^0| \ge 0.$$

There are two cases: (i) $|B^0| = 0$ and (ii) $|B^0| = 1$. In Case (i) we have $|B^-| - |B^+| = 0$ that is $|B^-| = |B^+|$ and the total number of points $b \in B$ with different i-th coordinate is $2|B^-|$ (or $2|B^+|$) that is this number should be even. In this case it is obvious that x_i is a median.

In Case (ii) we have $-1 \le |B^-| - |B^+| \le 1$. This means that there can be three different cases: (a) $|B^-| = |B^+| - 1$. In this case the number of points $b \in B$ with different i-th coordinate is even and x_i is a median coinciding with one of b_i ($b \in B$). (b) $|B^-| = |B^+|$. Then the number of points $b \in B$ with different i-th coordinates is odd and x_i is a median coinciding with the coordinate which is exactly in the middle. (c) $|B^-| = |B^+| + 1$. In this case the number of points $b \in B$ with different i-th coordinates is even and again x_i is a median coinciding with one of b_i ($b \in B$). □

It follows from Proposition 1 that in order to solve Problem (6) with the distance function d_1 one needs to find the median of the cluster B, that is in this case we do not need to explicitly solve the optimization problem (6).

Finally, consider Problem (6) with the distance function d_∞. In this case the objective function φ can be written as:

$$\varphi(x) = \frac{1}{|B|} \sum_{b \in B} \max_{i=1,\dots,n} |x_i - b_i|. \tag{7}$$

For each data point $b \in B$ consider the following function:

$$\theta_b(x) = \max_{i=1,\dots,n} |x_i - b_i| = \max_{i=1,\dots,n} \max\{x_i - b_i, b_i - x_i\}$$

and define the set:

$$R_b(x) = \{i \in \{1, \dots, n\} :\ |x_i - b_i| = \theta_b(x)\}.$$

For given $b \in B$ and $x \in \mathbb{R}^n$ introduce the sets

$$I_-(b, x) = \{i \in \{1, \dots, n\} :\ x_i < b_i\},\ I_+(b, x) = \{i \in \{1, \dots, n\} :\ x_i > b_i\},$$
$$I_0(b, x) = \{i \in \{1, \dots, n\} :\ x_i = b_i\}.$$

The condition $I_0(b,x)\bigcap R_b(x) \neq \emptyset$ implies that $R_b(x) = I_0(b,x) = \{1,\ldots,n\}$ and $\theta_b(x) = 0$. Then

$$\partial\theta_b(x) = \text{conv}\left\{-e^i, e^i,\ i \in \{1,\ldots,n\}\right\}.$$

Here $e^i \in \mathbb{R}^n$ is the i-th unit vector. If $I_0(b,x)\bigcap R_b(x) = \emptyset$, then $R_b(x) \subset I_-(b,x)\bigcup I_+(b,x)$. In this case

$$\partial\theta_b(x) = \text{conv}\left\{(0,\ldots,0,\text{sign}\,(x_i - b_i), 0\ldots,0) : i \in R_b(x)\right\},$$

where

$$\text{sign}\,(z) = \begin{cases} 0, & \text{if } z = 0; \\ -1, & \text{if } z < 0; \\ 1, & \text{if } z > 0. \end{cases}$$

Then the subdifferential of the function φ at x is:

$$\partial\varphi(x) = \frac{1}{|B|}\sum_{b\in B}\partial\theta_b(x).$$

The necessary and sufficient condition for a point x to be a minimum is: $0 \in \partial\varphi(x)$. It is easy to see that even for moderately large number of points in the set B one may have a huge number of extreme points in the subdifferential $\partial\varphi(x)$ and therefore, the computation of the whole subdifferential is not an easy task. Unlike d_1 and d_2^2 functions there is no explicit formula for finding the solution to the problem (6) with the function d_∞. In this case one can apply bundle-type algorithms of nonsmooth optimization [7] to solve (6). One can also apply smoothing techniques to approximate the function d_∞ by the smooth functions, to replace the problem (6) by the sequence of smooth optimization problems and then apply smooth optimization methods to solve them (see, for example, [5,42,43] for different smoothing functions).

4.2. **An algorithm for solving Problem (1).** The objective function (2) in Problem (1) is nonsmooth and nonconvex. Most algorithms for solving such problems are designed using the Clarke subdifferential. Next, we recall some definitions related to this subdifferential.

A function $\varphi : \mathbb{R}^n \to \mathbb{R}$ is called locally Lipschitz on $\mathbb{R}^n$ if for any bounded subset $X \subset \mathbb{R}^n$ there exists a constant $L > 0$ such that

$$|\varphi(x) - \varphi(y)| \leq L\|x - y\|$$

for all $x, y \in X$. Here $\|\cdot\|$ is the Euclidean norm. The generalized directional derivative $\varphi^0(x,u)$ of φ at x in the direction $u \in \mathbb{R}^n$ is [20]:

$$\varphi^0(x,u) := \limsup_{y\to x, \alpha\downarrow 0} \alpha^{-1}[\varphi(y + \alpha u) - \varphi(y)].$$

For locally Lipschitz functions the generalized directional derivative exists. The subdifferential $\partial\varphi(x)$ of the function φ at x is defined as [20]:

$$\partial\varphi(x) := \left\{v \in \mathbb{R}^n :\ \varphi^0(x,u) \geq \langle v, u\rangle,\ \forall u \in \mathbb{R}^n\right\}.$$

Here $\langle\cdot,\cdot\rangle$ stands for the inner product in $\mathbb{R}^n$. A vector $v \in \partial\varphi(x)$ is called a subgradient.

A function φ is called regular at $x \in \mathbb{R}^n$, if it is differentiable with respect to any direction $u \in \mathbb{R}^n$ at x and $\varphi'(x,u) = \varphi^0(x,u)$ for all $u \in \mathbb{R}^n$, where $\varphi'(x,u)$ is a derivative of the function φ at the point x in the direction u:

$$\varphi'(x,u) = \lim_{\alpha\downarrow 0}\alpha^{-1}[\varphi(x + \alpha u) - \varphi(x)].$$

Next, we study some properties of the objective function (2), where the function d is defined by one of the functions d_1, d_2^2 or d_∞.

Proposition 2. *The function f_k defined by (2) is locally Lipschitz for functions d_1, d_2^2 and d_∞.*

Proof. The proof follows from the facts that the functions d_1, d_2^2 and d_∞ are locally Lipschitz, functions represented as a minimum of the finite number of locally Lipschitz functions are locally Lipschitz and the sum of locally Lipschitz functions is also locally Lipschitz. $\square$

For each $a \in A$ consider the function:

$$\psi_a(x) = \min_{j=1,\dots,k} d(x^j, a), \ \ x = (x^1, \dots, x^k) \in \mathbb{R}^{kn}.$$

Let

$$R_a(x) = \left\{ j \in \{1, \dots, k\} : d(x^j, a) = \psi_a(x) \right\}.$$

The function ψ_a is directionally differentiable and

$$\psi_a'(x,u) = \min_{j \in R_a(x)} d'[(x^j, a), u^j], \ \ u = (u^1, \dots, u^k) \in \mathbb{R}^{kn}. \tag{8}$$

Here $d'[(x^j,a),u^j]$ is the directional derivative of the function d at the point $x^j \in \mathbb{R}^n$ in the direction $u^j, \ j \in R_a(x)$.

Proposition 3. *For d_1, d_2^2 and d_∞ the function f_k given by (2) is directionally differentiable at any $x = (x^1, \dots, x^k) \in \mathbb{R}^{kn}$ and*

$$f_k'(x,u) = \frac{1}{m} \sum_{a \in A} \min_{j \in R_a(x)} d'[(x^j, a), u^j], \ \ u = (u^1, \dots, u^k) \in \mathbb{R}^{kn}. \tag{9}$$

Proof. Since all functions d_1, d_2^2 and d_∞ are convex they are directionally differentiable at any x with respect to any direction $u \in \mathbb{R}^n$. Then the directional differentiability of the function f_k follows from its representation as a sum of minima functions. The expression (9) follows from the expression (8) for the directional derivative of the function ψ_a. $\square$

Corollary 1. *Assume that at a point $x = (x^1, \dots, x^k) \in \mathbb{R}^{kn}$ there exists $a \in A$ such that $|R_a(x)| \geq 2$ and $d'[(x^{j_1}, a), u] \neq d'[(x^{j_2}, a), u]$ for some $j_1, j_2 \in R_a(x)$ and $u \in \mathbb{R}^n$. Then the function f_k is not regular at x.*

Proof. The generalized directional derivative $f_k^0(x,u)$ of the function f_k at the point $x \in \mathbb{R}^{kn}$ in the direction $u = (u^1, \dots, u^k) \in \mathbb{R}^{kn}$ is given by

$$f_k^0(x,u) = \frac{1}{m} \sum_{a \in A} \max_{j \in R_a(x)} d'[(x^j, a), u^j].$$

This obviously implies that if the conditions of the corollary are satisfied then $f_k'(x,u) < f_k^0(x,u)$ for some $u \in \mathbb{R}^{kn}$. This completes the proof. $\square$

It is well-known that the Clarke subdifferential calculus with most widely used operations (summation, maximum) exists in the form of equalities only for regular functions. Furthermore, to get equalities more restrictive conditions on component functions are required for functions, defined using complex compositions. The calculus exists in the form of inclusions if these conditions are not satisfied which makes it not applicable in numerical algorithms. Corollary 1 implies that the Clarke subdifferential calculus, in general, cannot be applied to compute or estimate subgradients of the function f_k. In this situation derivative free algorithms or algorithms based on the approximate subgradients are better choice than algorithms that use exact subgradients.

The Discrete Gradient Method introduced in [6] uses discrete gradients to approximate subgradients and discrete gradients are computed using only values of a function. $n+1$ function evaluations are required to compute one discrete gradient, where n is the number of variables. In [13] the version of the discrete gradient method is introduced, where the number of function evaluations can be reduced significantly exploiting a special structure of the objective function such as the piecewise partial separability. Next, we recall definitions of partial and piecewise partial separable functions (see [13], for details) and then show that the function f_k is piecewise separable for distance(-like) functions d_1, d_2^2 and d_∞.

Definition 1. *A function $\varphi : \mathbb{R}^n \to \mathbb{R}$ is called partially separable iff there exists a family of $n \times n$ diagonal matrices $U_i,\ i = 1, \ldots, M, M \geq 1$ such that the function φ can be represented as follows:*

$$\varphi(x) = \sum_{i=1}^{M} \varphi_i(U_i x).$$

In other terms, the function φ is called partially separable if it can be represented as the sum of functions of a much smaller number of variables. If $M = n$ and $diag(U_i) = e^i,\ i = 1, \ldots, n$ then the function φ is separable.

Definition 2. *A function $\varphi : \mathbb{R}^n \to \mathbb{R}$ is said to be piecewise partially separable iff there exists a finite family of closed sets $D_1, \ldots, D_p$ such that $\bigcup_{i=1}^{p} D_i = \mathbb{R}^n$ and the function φ is partially separable on each set $D_i, i = 1, \ldots, p$.*

Next we will show that the function f_k is piecewise partially separable.

Proposition 4. *The function f_k defined by (2) is piecewise separable for the functions d_1, d_2^2 and d_∞.*

Proof. It is clear that functions d_1 and d_2^2 are separable and the function d_∞ is pecewise separable. Since the function $\psi_a(x) = \min_{j=1,\ldots,k} d(x^j, a), a \in A$ is represented as a minimum of functions each depending on a subset of variables, it is piecewise separable. Finally, the function f_k as a sum of piecewise separable functions $\psi_a, a \in A$ is piecewise separable itself by Proposition 7 in [13]. $\square$

It is shown in [13] that by exploiting piecewise partial separability it is possible to reduce the number of function evaluations by the discrete gradient method. The number of variables in Problem (1) is nk. This means that to compute one discrete gradient of the function f_k one needs its $nk + 1$ evaluations. However, piecewise separability of f_k allows us to apply the scheme from [13] to reduce this number to 2. Thus, the use of this scheme allows us to reduce $(nk + 1)/2$ times the number of function evaluations.

4.3. An algorithm for solving Problem (5).

The objective function (3) in Problem (5) is nonsmooth and nonconvex. The Clarke subdifferential of this function can be used to design algorithms for solving Problem (5). However, similar to the function (2), the computation of the subdifferential of the function (3) is not always an easy task.

Proposition 5. *The function $\bar{f}_l$ defined by (3) is locally Lipschitz for functions d_1, d_2^2 and d_∞.*

Proof. The proof follows from the facts that the functions d_1, d_2^2 and d_∞ are locally Lipschitz, functions represented as a minimum of a constant and a locally Lipschitz function are locally Lipschitz and the sum of locally Lipschitz functions is also locally Lipschitz. $\square$

It follows from Proposition 5 that the function $\bar{f}_l$ is subdifferentiable. For a given $y \in \mathbb{R}^n$ by introducing the sets

$$\bar{A}_1(y) = \{a \in A : \; r^a_{l-1} > d(y,a)\}, \;\; \bar{A}_2(y) = \{a \in A : \; r^a_{l-1} = d(y,a)\},$$
$$\bar{A}_3(y) = \{a \in A : \; r^a_{l-1} < d(y,a)\}$$

one can rewrite the function $\bar{f}_l$ as follows:

$$\bar{f}_l(y) = \frac{1}{m}\left(\sum_{a\in\bar{A}_1(y)} d(y,a) + \sum_{a\in\bar{A}_2(y)} \min\{r^a_{l-1}, d(y,a)\} + \sum_{a\in\bar{A}_3(y)} r^a_{l-1}\right). \tag{10}$$

Proposition 6. *For distance(-like) functions d_1, d_2^2 and d_∞ the function $\bar{f}_l$ given by (3) is directionally differentiable at any $y \in \mathbb{R}^n$ and*

$$\bar{f}'_l(y,u) = \frac{1}{m}\left(\sum_{a\in\bar{A}_1(y)} d'[(y,a),u] + \sum_{a\in\bar{A}_2(y)} \min\{0, d'[(y,a),u]\}\right), \; u \in \mathbb{R}^n. \tag{11}$$

Proof. The functions d_1, d_2^2 and d_∞ are convex and therefore, they are directionally differentiable at any y in any direction $u \in \mathbb{R}^n$. Then the directional differentiability of the function $\bar{f}_l$ follows from its representation as a sum of minima functions. The expression (11) follows from the expression (10) for the function $\bar{f}_l$ at the point y. $\square$

Corollary 2. *If the set $\bar{A}_2(y) = \emptyset$ at a point $y \in \mathbb{R}^n$, then the function $\bar{f}_l$ is regular at this point.*

Proof. The proof follows from the representation (11) for the directional derivative. $\square$

Corollary 3. *If at a point $y \in \mathbb{R}^n$ there exist at least one $a \in \bar{A}_2(y)$ such that $d'[(y,a),u] \neq 0$ for some $u \in \mathbb{R}^n$, then the function $\bar{f}_l$ is not regular at this point.*

Proof. The generalized directional derivative of the function $\bar{f}_l$ at the point $y \in \mathbb{R}^n$ in the direction $u \in \mathbb{R}^n$ is:

$$\bar{f}^0_l(y,u) = \frac{1}{m}\left(\sum_{a\in\bar{A}_1(y)} d'[(y,a),u] + \sum_{a\in\bar{A}_2(y)} \max\{0, d'[(y,a),u]\}\right).$$

Then it follows from the conditions of the corollary and (11) that $\bar{f}'_l(y,u) < \bar{f}^0_l(y,u)$ for some $u \in \mathbb{R}^n$. This completes the proof. $\square$

Next we prove that the function $\bar{f}_l$ is piecewise separable.

Proposition 7. *The function $\bar{f}_l$ given by (3) is piecewise separable for functions d_1, d_2^2 and d_∞.*

Proof. The functions d_1 and d_2^2 are separable and the function d_∞ is piecewise separable. Therefore for each $a \in A$ the function $\min\{r^a_{l-1}, d(y,a)\}$ is piecewise separable for all functions d_1, d_2^2 and d_∞. Then piecewise separability of the function $\bar{f}_l$ follows from the fact that the sum of piecewise separable functions is also piecewise separable [13]. $\square$

Since the function $\bar{f}_l$ is not regular, the Clarke subdifferential calculus, in general, cannot be used to calculate its subgradients. Therefore, we apply the discrete gradient method to solve Problem (5). Since $\bar{f}_l$ is piecewise separable, we apply the simplified version of this method from [13]. This version of the method requires $(n+1)/2$ times less function evaluations than the discrete gradient method itself.

5. **Computation of starting points.**

The clustering is a global optimization problem. To improve the accuracy of local search algorithms in solving such problems it is crucial to develop a special procedure to generate starting points. For example, such procedures were introduced to improve the performance of the k-means algorithm [2,19]. In this section, first we describe an algorithm to generate starting points for the auxiliary clustering problem (5) and then use these points to construct starting cluster centers for the problem (1). The similar algorithms were also developed in [10,32].

Given the solution $x^1,\dots,x^{l-1},\ l>1$ to the $(l-1)$-clustering problem one can divide the whole space $\mathbb{R}^n$ into two subsets as follows:

$$S_1 = \left\{ y \in \mathbb{R}^n :\ d(y,a) \geq r^a_{l-1},\ \forall a \in A \right\},$$

$$S_2 = \left\{ y \in \mathbb{R}^n : \exists a \in A \text{ such that } d(y,a) < r^a_{l-1} \right\}.$$

The set S_1 contains all points from $\mathbb{R}^n$ that do not attract any point from the set A and the set S_2 contains points which attract at least one point from A. Note that $S_1 \bigcup S_2 = \mathbb{R}^n$ and $S_1 \bigcap S_2 = \emptyset$. Moreover,

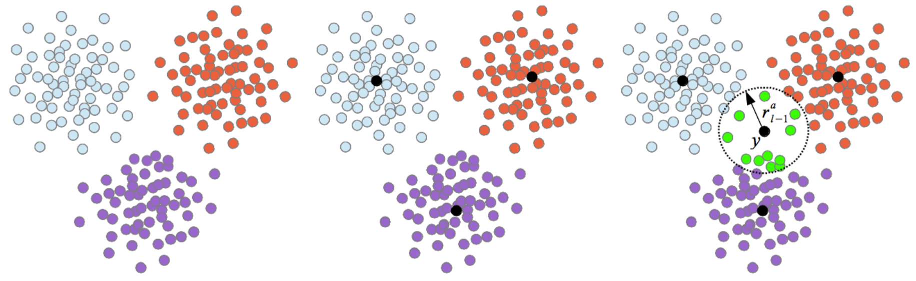
$$\bar{f}_l(y) = f_{l-1}(x^1,\dots,x^{l-1}) = \frac{1}{m}\sum_{a\in A} r^a_{l-1} = f^*_{l-1},\ \forall y \in S_1,$$

that is the function $\bar{f}_l$ is constant on the set S_1 and according to (4) points from this set are its global maximizers. In general, a local search method will terminate at any of these points and therefore, they cannot be considered as starting points to solve the problem (5).

Take any $y \in S_2$. The set A can be divided into two subsets as follows:

$$B_1(y) = \left\{ a \in A :\ d(y,a) \geq r^a_{l-1} \right\},\ B_2(y) = \left\{ a \in A :\ d(y,a) < r^a_{l-1} \right\}.$$

The set $B_2(y)$ contains data points which are closer to the point y than their cluster centers. This set is illustrated in Figure 1.



(a) Three clusters (b) Three clusters with centers (c) The set $B_2(y)$ (green balls)

FIGURE 1. Illustration of the set $B_2(y)$.

The set $B_2(y) \neq \emptyset$, $B_1(y) \bigcap B_2(y) = \emptyset$ and $A = B_1(y) \bigcup B_2(y)$. Then

$$\bar{f}_l(y) = \frac{1}{m} \left(\sum_{a \in B_1(y)} r^a_{l-1} + \sum_{a \in B_2(y)} d(y,a) \right).$$

The difference $v_l(y)$ between the values $f_{l-1}(x^1, \dots, x^{l-1})$ and $\bar{f}_l(y)$ is:

$$v_l(y) = \sum_{a \in A} \max \left\{ 0, r^a_{l-1} - d(y,a) \right\}. \tag{12}$$

To compute starting points for the problem (5) we use points $a \in A \setminus S_1 \subset S_2$. Consider the number:

$$v^1_{max} = \max_{a \in A \setminus S_1} v_l(a). \tag{13}$$

A data point $\bar{a} \in A$, satisfying the condition $v_l(\bar{a}) = v^1_{max}$, provides the largest decrease of the cluster function f_l comparing with the value f^*_{l-1}:

$$\bar{a} = \arg\max_{a \in A} \left[f^*_{l-1} - f_l(x^1, \dots, x^{l-1}, a) \right].$$

Let $\gamma_1 \in [0,1]$ be a given number. Define the following subset of A:

$$\bar{A}_1 = \left\{ a \in A \setminus S_1 : \; v_l(a) \geq \gamma_1 v^1_{max} \right\}. \tag{14}$$

The set $\bar{A}_1$ contains points $a \in A \setminus S_1$ which as the l-th cluster center provide the sufficient decrease of the cluster function f_l comparing with the value f^*_{l-1}. If $\gamma_1 = 1$, then A_1 contains points with the largest decrease and if $\gamma_1 = 0$ then $\bar{A}_1 = A \setminus S_1$.

For each $a \in \bar{A}_1$ compute the set $B_2(a)$ and its center $c(a)$. We replace points $a \in \bar{A}_1$ by the points $c(a)$, because the center $c(a)$ is the better representative of the set $B_2(a)$ than the point a. Then we get

$$\bar{C}_1 = \{ c \in \mathbb{R}^n : \; \exists a \in \bar{A}_1 \text{ such that } c = c(a) \}.$$

Applying (12) we compute $v_l(c)$ for each $c \in \bar{C}_1$ and then find:

$$v^2_{max} = \max_{c \in \bar{C}_1} v_l(c). \tag{15}$$

Let $\gamma_2 \in [0,1]$ be a given number. Define the following subset of $\bar{C}_1$:

$$\bar{A}_2 = \left\{ c \in \bar{C}_1 : \; v_l(c) \geq \gamma_2 v^2_{max} \right\}. \tag{16}$$

Points from the set $\bar{A}_2$ are considered as starting points for solving the auxiliary clustering problem (5). Applying the discrete gradient method (see Subsection 4.3) Problem (5) is solved starting from points $a \in \bar{A}_2$. Then we get the set $\bar{A}_3$ of solutions of Problem (5). Since the local method can arrive to the same solution starting from different points we remove from the set $\bar{A}_3$ points which are sufficiently close to each other keeping only one of them. Sufficiently close points can be defined using some predefined tolerance. It is clear that $\bar{A}_3 \neq \emptyset$. Next, we define

$$\bar{f}^{min}_l = \min_{y \in \bar{A}_3} \bar{f}_l(y). \tag{17}$$

Let $\gamma_3 \in [1, \infty)$ be a given number. Compute the following set:

$$\bar{A}_4 = \left\{ y \in \bar{A}_3 : \bar{f}_l(y) \leq \gamma_3 \bar{f}^{min}_l \right\}. \tag{18}$$

If $\gamma_3 = 1$ then $\bar{A}_4$ contains stationary points of the problem (5) with the lowest value $\bar{f}^{min}_l$. If γ_3 is sufficiently large then $\bar{A}_4 = \bar{A}_3$. Summarizing all steps we can design the following algorithm for computing the set $\bar{A}_4$.

Algorithm 1. Computation of starting points.

Step 0. (Initialization). Select $\gamma_1, \gamma_2 \in [0,1]$ and $\gamma_3 \in [1, \infty)$.

Step 1. Compute v^1_{max} using (13) and the set $\bar{A}_1$ using (14).

Step 2. Compute v^2_{max} using (15) and the set $\bar{A}_2$ using (16).

Step 3. Using points from the set $\bar{A}_2$ as starting points compute the set $\bar{A}_3$ of solutions of the auxiliary clustering problem (5).

Step 4. Compute f^{min}_k using (17) and the set $\bar{A}_4$ using (18).

6. **The incremental clustering algorithm and its implementation.** The incremental algorithm for solving Problem (1) proceeds as follows.

Algorithm 2. An incremental clustering algorithm.

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 $l > k$ then stop. The k-partition problem has been solved.

Step 3. (Computation of the starting points). Apply Algorithm 1 to compute the set $\bar{A}_4$ of starting points for the l-th cluster center.

Step 4. (Refinement of all cluster centers). For each $\bar{y} \in \bar{A}_4$ select $(x^1, \ldots, x^{l-1}, \bar{y})$ as a starting point to solve the l-partition problem (1) (when $k = l$). Let $(y^1(\bar{y}), \ldots, y^l(\bar{y}))$ be a solution to this problem and $f_{l\bar{y}} = f_l(y^1(\bar{y}), \ldots, y^l(\bar{y}))$ be a value of the cluster function at this solution.

Step 5. (Computation of the best solution). Compute

$$f^{min}_l = \min_{\bar{y} \in \bar{A}_4} f_{l\bar{y}}$$

and the set of best solutions

$$C = \left\{(y^1(\bar{y}), \ldots, y^l(\bar{y})) : f_{l\bar{y}} = f^{min}_l\right\}.$$

Step 6. (Solution to the l-partition problem). Take any $(y^1(\bar{y}), \ldots, y^k(\bar{y})) \in C$, set $x^j := y^j(\bar{y}),\ j = 1, \ldots, l$ as a solution to the l-th partition problem and go to Step 2.

Remark 1. To date, incremental algorithms were designed for solving the minimum sum-of-squares clustering problems [4, 10, 14, 15, 31, 32]. In this paper, we extend them to solve clustering problems with the similarity measures based on the L_1 and L_∞ norms.

Remark 2. One can see that Algorithm 2 in addition to the k-partition problem solves also all intermediate l-partition problems with $l = 1, \ldots, k-1$. In general, Algorithm 2 can find only stationary points of the clustering problem (1). However, the use of the special procedure to generate good starting points allows to find either global or near global solutions to this problem which will be confirmed by the computational results.

The most important steps in Algorithm 2 are Steps 3 and 4. Other steps of this algorithm are easy to implement. In Step 3 we apply Algorithm 1 to find the set of starting points for the l-th cluster center. Algorithm 1 contains three parameters, namely, γ_1, γ_2 and γ_3. The choice of these parameters depends on the size of a data set.

One can see from (14) that if γ_1 is close to 0, then the set $\bar{A}_1$ will contain most of data points which may lead to a large number of starting points and make the algorithm very time-consuming. Results presented in [10] show that, in general, $\gamma_2 \geq \gamma_1$ and the values of these parameters should converge to 1 as the number of data points increase. Such a choice does not deteriorate the solution of the clustering problem but decreases computational time significantly. Since the difference between the value $\bar{f}_l^{min}$ of the auxiliary clustering problem (5) and the clustering problem (1) is not expected to be large one can choose the parameter γ_3 from $[1,2]$. In the implementation of Algorithm 2 the parameters are chosen as follows:

1. $\gamma_1 = 0.4, \gamma_2 \in [0.5, 0.6]$ and $\gamma_3 = 1.1$ in data sets with the number of data points $m \leq 200$;
2. $\gamma_1 = 0.6, \gamma_2 \in [0.8, 0.9]$ and $\gamma_3 = 1.05$ in data sets with $200 < m \leq 2500$;
3. $\gamma_1 \in [0.7, 0.8], \gamma_2 \in [0.85, 0.9]$ and $\gamma_3 = 1.05$ in data sets with $2500 < m \leq 20000$;
4. $\gamma_1 = [0.85, 0.97], \gamma_2 \in [0.97, 0.995]$ and $\gamma_3 = 1.025$ in data sets with $m > 20000$.

In order to determine the neighboring points from the set $\bar{A}_3$ we use the following tolerance:

$$\varepsilon = \frac{10^{-4} f_1^{min}}{l}$$

where f_1^{min} is the optimal value of the cluster function f_k when $k = 1$ and l is the number of clusters. If the distance between two points in $\bar{A}_3$ is less than ε then we remove one of these points (any) from the set.

Since the proposed algorithm is incremental, data points which are close to previous cluster centers are not considered as candidates to be starting points. In order to identify such points we apply a scheme already discussed in [14]. The discrete gradient method is applied to solve problems (1) and (5). The proposed algorithm: INCA - Incremental Nonsmooth Clustering Algorithm was implemented in Fortran 95 and compiled using *gfortran*, a GNU Fortran compiler.

7. Computational results.

The INCA was tested and compared with other clustering algorithms using 12 real-world data sets. Numerical experiments were performed on an Intel(R) Core^{TM} i5-3470S CPU 2.90 GHz and RAM 8GB. The selected data sets satisfy the following conditions: 1) they have no missing values; 2) all attributes are numeric (either integer or continuous); 3) the number of points in data sets varies from very small (89) to large (up to 85900); 4) the number of attributes in data sets varies from small (2) to relatively large (up to 20). The use of such data sets allows one to get close to comprehensive picture on the performance of the algorithm. The brief description of data sets is given in Table 1 (the dash line in this table indicates that the corresponding data sets have no class label). The detailed description of the TSPLIB1060, TSPLIB3038, D15112 and Pla85900 data sets can be found in [36]. Other data sets are from [3]. Notice that three attributes were removed from KEGG Metabolic Relation Network data set since among them two attributes have only 0 value and the third attribute is 0 for all points but one.

7.1. Optimal values for cluster functions.

We compute up to 10 clusters in small data sets (Bavaria postal 1, Bavaria postal 2 and Iris Plant), up to 20 clusters in medium size data sets (Breast Cancer, TSPLIB1060 and Image segmentation) and up to 25 clusters in large data sets (TSPLIB3038, Page Blocks, EEG Eye State, D15112, KEGG Metabolic Relation Network and Pla85900).

Tables 2-3 present optimal values of the cluster function f_k obtained by the proposed algorithm for different similarity measures and the number of clusters k. Note that these values are multiplied by m - the number of points in a data set.

TABLE 1. The brief description of data sets

Data sets	Number of instances	Number of attributes	Number of classes
Bavaria postal 1	89	3	-
Bavaria postal 2	89	4	-
Fisher's Iris Plant	150	4	3
Breast Cancer	683	9	2
TSPLIB1060	1060	2	-
Image Segmentation	2310	19	7
TSPLIB3038	3038	2	-
Page Blocks	5473	10	5
EEG Eye State	14980	14	2
D15112	15112	2	-
KEGG Metabolic Relation Network	53413	20	-
Pla85900	85900	2	-

From these results one can see that in all cases, except Iris Plant data set, the values of the cluster function with d_∞ are smallest among all three functions.

TABLE 2. Results for small and medium size data sets

k	d_1	d_2^2	d_∞	d_1	d_2^2	d_∞	d_1	d_2^2	d_∞
	Bavaria postal 1			Bavaria postal 2			Iris Plant		
	$\times 10^6$	$\times 10^{10}$	$\times 10^6$	$\times 10^6$	$\times 10^{10}$	$\times 10^6$	$\times 10^2$	$\times 10^2$	$\times 10^2$
2	4.0249	60.2547	3.9940	1.8600	5.2192	0.9456	2.1670	1.5235	0.9715
3	2.8284	29.4507	2.7892	1.2607	1.7399	0.6594	1.5920	0.7885	0.7420
4	2.1982	10.4561	2.1690	0.9633	0.7559	0.5210	1.3650	0.5726	0.6460
5	1.7208	5.9762	1.6948	0.7872	0.5442	0.4221	1.2460	0.4645	0.5860
6	1.3003	3.5909	1.2766	0.6736	0.3226	0.3459	1.1530	0.3904	0.5300
7	1.0704	2.1983	1.0368	0.5659	0.2215	0.2946	1.0620	0.3430	0.4915
8	0.8511	1.3385	0.8183	0.5097	0.1705	0.2550	1.0010	0.2999	0.4640
9	0.7220	0.8424	0.6993	0.4688	0.1401	0.2360	0.9540	0.2779	0.4435
10	0.6037	0.6447	0.5828	0.4340	0.1181	0.2173	0.9070	0.2583	0.4245
	Breast Cancer			TSPLIB1060			Image Segmentation		
	$\times 10^4$	$\times 10^4$	$\times 10^4$	$\times 10^7$	$\times 10^9$	$\times 10^6$	$\times 10^6$	$\times 10^7$	$\times 10^5$
2	0.6401	1.9323	0.1831	0.3864	9.8319	2.6809	0.5192	3.5606	1.4929
3	0.5702	1.6256	0.1607	0.3139	6.7058	2.1508	0.4160	2.7416	1.3284
5	0.5165	1.3707	0.1460	0.2310	3.7915	1.6546	0.3400	1.7143	1.1081
7	0.4651	1.2031	0.1372	0.1976	2.7039	1.3754	0.2892	1.3473	0.9408
10	0.4270	1.0212	0.1278	0.1563	1.7553	1.1048	0.2575	0.9967	0.8170
12	0.4068	0.9480	0.1234	0.1378	1.4507	1.0033	0.2401	0.8477	0.7635
15	0.3872	0.8711	0.1172	0.1198	1.1219	0.8827	0.2188	0.6556	0.6966
18	0.3707	0.8068	0.1112	0.1080	0.9004	0.7906	0.2021	0.5630	0.6481
20	0.3614	0.7698	0.1081	0.1015	0.7925	0.7378	0.1942	0.5137	0.6200

7.2. **CPU time and number of distance calculations.** CPU time used by the INCA for all similarity measures are given in Tables 4 and 5. The following conclusions can be made based on these results:

1. The algorithm requires the largest CPU time with the function d_∞ in all data sets except Bavaria postal 1 and TSPLIB1060 and the least CPU time with

TABLE 3. Results for large data sets

k	d_1	d_2^2	d_∞	d_1	d_2^2	d_∞	d_1	d_2^2	d_∞
	TSPLIB3038			Page Blocks			D15112		
	$\times 10^6$	$\times 10^9$	$\times 10^6$	$\times 10^7$	$\times 10^{10}$	$\times 10^6$	$\times 10^8$	$\times 10^{11}$	$\times 10^8$
2	3.7308	3.1688	2.5651	0.8414	5.7937	4.1746	0.8860	3.6840	0.6109
3	3.0056	2.1763	2.1221	0.6747	3.3134	3.4309	0.6908	2.5324	0.4896
5	2.2551	1.1982	1.5576	0.4882	1.3218	2.4671	0.4998	1.3271	0.3619
10	1.5508	0.5634	1.0738	0.3152	0.4533	1.4446	0.3618	0.6489	0.2524
15	1.2295	0.3560	0.8592	0.2555	0.2495	1.1784	0.2930	0.4324	0.2065
20	1.0597	0.2672	0.7379	0.2200	0.1672	1.0160	0.2501	0.3218	0.1768
25	0.9441	0.2145	0.6627	0.2004	0.1203	0.9020	0.2241	0.2530	0.1583
	EEG Eye State			Pla85900			Relation Network		
	$\times 10^7$	$\times 10^8$	$\times 10^6$	$\times 10^{10}$	$\times 10^{15}$	$\times 10^{10}$	$\times 10^7$	$\times 10^8$	$\times 10^6$
2	0.5289	8178.1381	1.5433	2.0656	3.7491	1.4533	0.3586	11.3853	1.9821
3	0.4197	1833.8806	0.9049	1.6262	2.2806	1.1434	0.2800	4.9006	1.5112
5	0.2944	1.3386	0.5183	1.2587	1.3397	0.8712	0.2095	1.8837	1.0549
10	0.2191	0.4567	0.3947	0.8950	0.6829	0.6218	0.1459	0.6352	0.6667
15	0.1965	0.3500	0.3562	0.7335	0.4625	0.5082	0.1231	0.3512	0.5114
20	0.1827	0.2899	0.3292	0.6374	0.3517	0.4443	0.1108	0.2654	0.4440
25	0.1736	0.2596	0.3129	0.5693	0.2827	0.3981	0.1011	0.1946	0.4013

the function d_2^2 in all data sets. The clustering problem with the function d_∞ is the most complex among all and the discrete gradient method requires a large number of approximate subgradient evaluations to find search directions in this problem. On the other side, the problem with the function d_2^2 is easiest as this function is smooth. In this case the optimization method does not require a large number of approximate subgradient evaluations to find search directions.

2. CPU time required by the algorithm depends more strongly on the number of attributes than on the number of data points. This claim is confirmed by comparing results for data sets with the similar number of data points and significantly different number of attributes: Image Segmentation and TSPLIB3038; D15112 and EGE Eye State; Pla85900 and KEGG Metabolic Relation Network. The comparison shows that the algorithm can become time-consuming in large data sets with the large number of attributes. In such data sets the size of the optimization problem increases rapidly as the number of clusters increase.
3. For all similarity measures CPU time required at each iteration of the incremental algorithm, in general, is more than that of required at the previous iterations. This is due to the fact that the size of the optimization problem for finding all cluster centers increases at each iteration of the incremental algorithm.

The dependence of the number of distance(-like) function evaluations on the number of clusters for all data sets and similarity measures are displayed in Figure 2. One can see that the algorithm requires less number of distance evaluations with the function d_2^2 than with other 2 functions. Results for d_1 and d_∞ are mixed.

There is no strong correlation between the CPU time and the number of distance evaluations. For example, in data sets D15112 and PLa85900 the algorithm with d_1 and d_∞ uses similar number of distance evaluations, however it requires significantly

Table 4. CPU time in small and medium size data sets

k	d_1	d_2^2	d_∞	d_1	d_2^2	d_∞	d_1	d_2^2	d_∞
	Bavaria postal 1			Bavaria postal 2			Iris Plant		
2	0.03	0.00	0.03	0.03	0.00	0.03	0.02	0.00	0.02
3	0.05	0.02	0.05	0.03	0.00	0.03	0.12	0.00	0.27
4	0.05	0.02	0.05	0.05	0.02	0.05	0.20	0.02	0.70
5	0.20	0.03	0.14	0.12	0.02	0.20	0.42	0.05	1.05
6	0.22	0.05	0.16	0.16	0.03	0.25	0.56	0.11	1.42
7	0.25	0.05	0.19	0.20	0.05	0.42	0.70	0.19	2.12
8	0.25	0.08	0.19	0.28	0.09	0.59	0.94	0.22	2.98
9	0.41	0.11	0.27	0.31	0.14	1.00	1.29	0.51	4.13
10	0.50	0.14	0.30	0.34	0.17	1.33	2.26	0.80	4.90
	Breast Cancer			TSPLIB1060			Image Segmentation		
2	1.45	0.03	0.50	0.09	0.05	0.11	2.28	0.33	27.69
3	5.93	0.22	1.95	0.30	0.08	0.39	5.38	0.89	61.67
5	12.20	1.11	9.94	1.15	0.17	1.29	13.37	3.53	426.55
7	15.58	2.45	14.52	3.17	0.53	4.91	27.67	8.53	638.29
10	22.45	3.42	17.49	7.36	0.86	10.64	157.87	14.82	1144.10
12	25.05	5.91	32.53	9.41	1.53	17.89	210.59	23.13	1539.23
15	26.29	8.89	55.13	18.19	2.22	26.80	360.63	34.02	1903.06
18	28.92	12.37	85.97	30.81	3.40	35.66	508.86	41.34	2284.46
20	30.47	15.09	96.22	47.22	4.84	57.30	711.24	62.63	2866.97

Table 5. CPU time in large data sets

k	d_1	d_2^2	d_∞	d_1	d_2^2	d_∞	d_1	d_2^2	d_∞
	TSPLIB3038			Page Blocks			D15112		
2	0.33	0.20	0.41	2.67	0.45	1.22	4.80	4.48	6.13
3	0.90	0.47	0.95	6.51	1.34	7.52	8.80	8.50	12.62
5	2.34	0.87	1.56	30.75	2.84	54.16	15.37	14.71	23.51
10	14.29	2.26	14.68	113.05	14.01	193.61	39.13	32.26	54.49
15	43.34	4.48	33.79	193.38	29.05	523.04	73.79	58.34	107.72
20	107.33	8.31	63.73	739.82	63.68	787.13	121.87	92.51	176.70
25	163.46	15.29	155.72	969.64	95.22	1085.21	168.96	136.61	274.31
	EEG Eye State			Pla85900			Relation Network		
2	1.26	0.89	2.57	79.47	77.69	108.03	95.22	4.93	71.46
3	2.20	1.61	3.96	156.24	154.13	216.56	208.09	18.50	119.70
5	9.28	3.79	16.04	311.22	305.59	431.59	422.79	52.45	283.08
10	44.80	65.86	79.89	710.71	697.09	989.53	949.52	186.80	931.51
15	115.74	148.62	208.57	1138.17	1154.27	1566.55	1789.21	514.96	1963.96
20	252.27	243.49	415.32	1574.80	1575.67	2177.26	2687.01	787.71	3546.57
25	414.92	354.20	764.83	2041.79	2025.19	2830.90	4547.85	1197.28	5854.08

less CPU time with d_1 than with d_∞. This means that the discrete gradient method requires more or more time for solving the subproblem for finding search directions as the number of clusters increase.

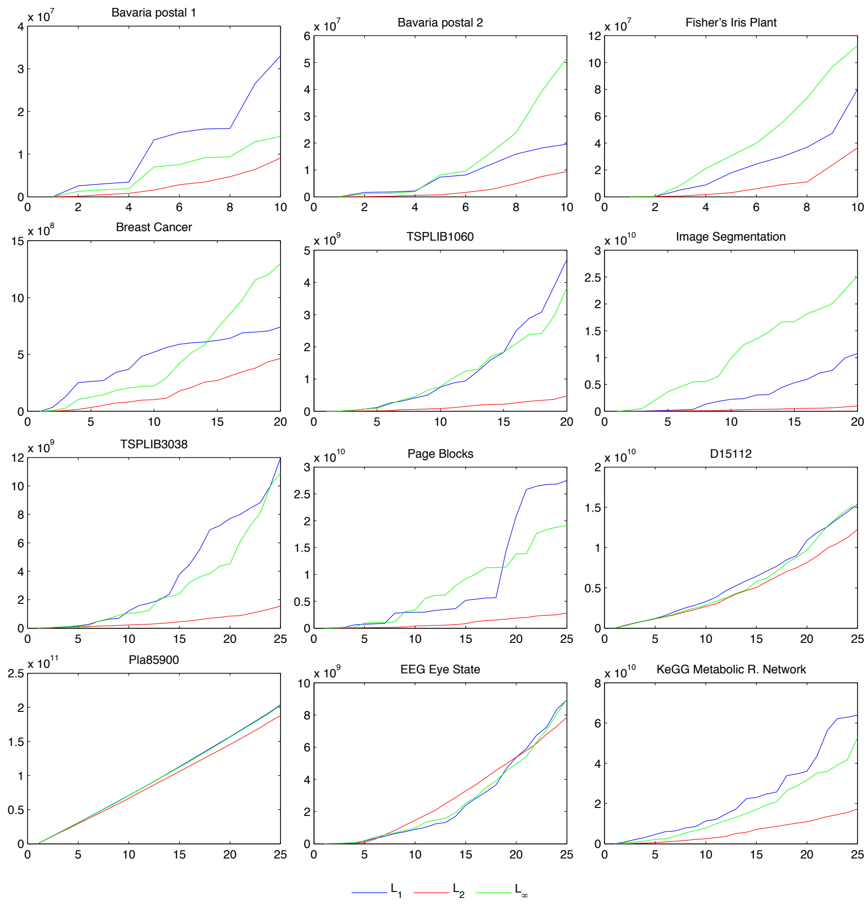


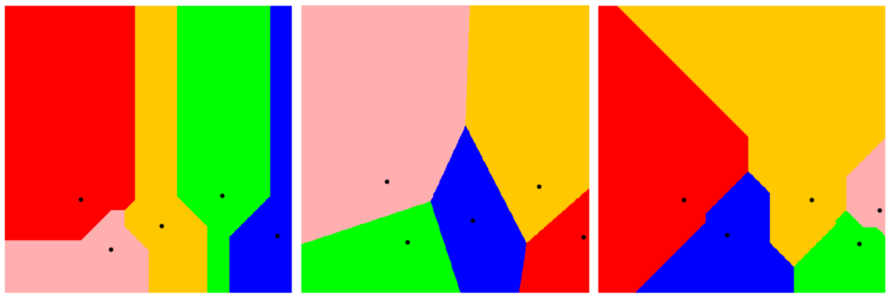
FIGURE 2. The number of distance evaluations vs the number of clusters

7.3. **Visualization of results.** Voronoi diagrams are used to visualize results obtained by the INCA in two data sets: TSPLIB1060 and TSPLIB3038. The software from [38] is used for this purpose. Figures 3 and 4 present diagrams for five clusters in both data sets. One can see that cluster structures for different similarity measures are different in both data sets, although the distributions of cluster centers for d_1 and d_2^2 functions in TSPLIB1060 data set are quite similar.

7.4. **Results for purity in data sets with class labels.** In order to determine which similarity measure provides better approximation of classes in a data set with class labels one can use the notion of the *cluster purity*. The purity P_i of the cluster $i,\ i = 1,\dots,k$ is defined as follows [15]:

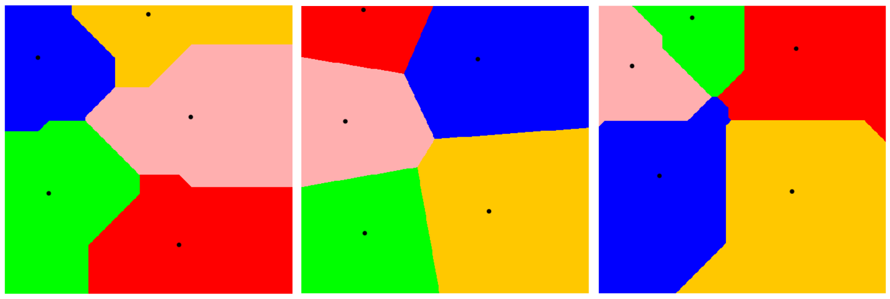
$$P_i = \frac{1}{n_i} \max_{j=1,\dots,l} \ n_i^j.$$

In this expression n_i is the number of points in the cluster $i,\ i = 1,\dots,k, n_i^j$ is the number of points in the cluster i that belong to the true class j and l is the number



(a) L_1-norm (b) L_2-norm (c) L_∞-norm

FIGURE 3. Visualization of clusters for TSPLIB1060 data set.



(a) L_1-norm (b) L_2-norm (c) L_∞-norm

FIGURE 4. Visualization of clusters for TSPLIB3038 data set.

of true classes. The *total purity* P_t for the whole data set A is computed as:

$$P_t = \frac{1}{m} \sum_{i=1}^{k} \max_{j=1,\dots,l} n_i^j.$$

Among all data sets used in our experiments five contain class labels: Iris Plant, Breast Cancer, Image Segmentation, Page Blocks and EEG Eye State. Results are illustrated in Figure 5. Here we give the total purity in % that is P_t is multiplied by 100. One can see that different similarity measures generate very similar results in the Iris Plant and Breast Cancer data sets. There are three classes in Iris Plant data set and one of them is linearly separable from other two. Breast Cancer data set contains two classes and they are almost linearly separable. This can explain why purity results for all distance functions are similar in these data sets. In Image Segmentation and Page Blocks data sets the distance function d_1 achieves the best results for the purity. In EEG State Eye data set the distance function d_∞ achieves the best result and the function d_1 gets the result which is very close to the best one but with significantly less number of clusters. All these three data sets contain classes which are not linearly separable. Results demonstrate that overall the algorithm with the function d_1 provides better approximation of classes than other 2 functions in data sets with not well separated classes.

7.5. **Comparison of algorithms.** In this subsection using numerical results we compare the INCA with the X-means algorithm [33]. Similar to the INCA this

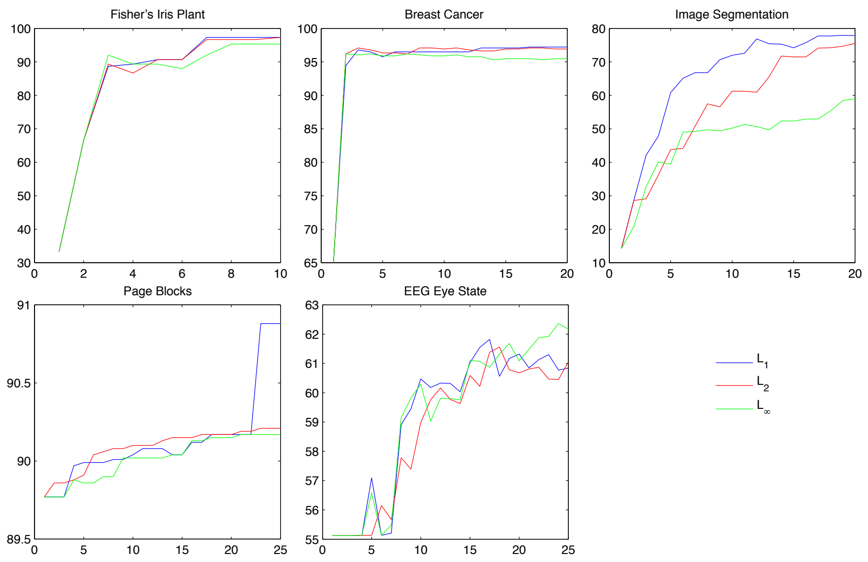


FIGURE 5. The purity vs the number of clusters

algorithm uses the special procedure for finding starting cluster centers. We present the comparison with the similarity measures based on the d_1 and d_∞ distance functions. In order to have a fair comparison we run the X-means algorithm using CPU time three times of that required by the INCA for the same number of clusters. The comparison is based on the error $E_k, k \geq 2$ computed as:

$$E_k = \frac{\bar{f}_k - f_k^{best}}{f_k^{best}} \times 100\%$$

where $\bar{f}_k$ is the value of the k-clustering problem obtained by an algorithm and f_k^{best} is the best known value of this problem.

We report results only for four small and medium size data sets because the INCA considerably outperforms the X-means algorithm in other data sets. Results, presented in Table 6, clearly demonstrate that the INCA is far more efficient than the X-means algorithm for solving clustering problems where the similarity measure is defined using d_1 and d_∞ functions.

7.6. **Robustness of algorithms to outliers.** In order to demonstrate the robustness of algorithms with respect to outliers we consider one 2-dimensional synthetic data set containing three well-separated clusters with 33, 40 and 47 data points and 4 small groups of outliers (with $2, 2, 2$ and 3 data points). This data set is illustrated in Figure 6.

The proposed algorithm with both the L_1 and L_∞ norms at the 7-th iteration computes 7 clusters and two of them contain only outliers. In both cases the algorithm classifies four outlier points on the left hand in one cluster and five outlier points on the right hand side to another cluster. The algorithm divides the largest cluster (with 47 data points) into three clusters and generate other two clusters as is. This means that for this data set, the proposed algorithm with both the L_1 and

Table 6. Comparison of algorithms using d_1 and d_∞ functions

k	d_1		d_∞		k	d_1		d_∞	
	X-means	INCA	X-means	INCA		X-means	INCA	X-means	INCA
	Bavaria postal 1					Bavaria postal 2			
2	44.21	0.00	44.58	0.00	2	13.31	0.00	15.95	0.00
3	34.44	0.00	111.42	0.00	3	15.29	0.00	23.33	0.00
4	48.73	0.00	63.68	0.00	4	27.51	0.00	30.53	0.00
5	102.81	0.00	102.90	0.00	5	14.30	0.00	45.22	0.00
6	149.68	0.00	166.86	0.00	6	10.24	0.00	72.81	0.00
7	148.86	0.00	226.41	0.00	7	20.39	0.00	97.27	0.00
8	188.37	0.00	312.47	0.00	8	32.67	0.00	122.49	0.00
9	338.74	0.00	370.37	0.00	9	39.57	0.00	132.80	0.00
10	404.43	0.00	466.69	0.00	10	42.22	0.00	149.03	0.00
	Iris Plants					Breast Cancer			
2	2.05	0.00	1.01	0.00	2	14.46	0.00	5.60	0.00
3	2.07	0.00	6.45	0.00	3	15.35	0.00	14.25	0.00
4	12.42	0.00	12.02	0.00	5	15.11	0.00	9.26	0.00
5	5.34	0.00	10.58	0.00	7	20.19	0.00	10.40	0.00
6	7.15	0.00	30.88	0.00	10	18.15	0.00	14.19	0.00
7	11.85	0.00	21.67	0.00	12	19.75	0.00	12.80	0.00
8	17.12	0.00	26.90	0.00	15	18.75	0.00	14.41	0.00
9	21.59	0.00	31.06	0.00	18	21.63	0.00	15.84	0.00
10	23.72	0.00	28.98	0.00	20	22.22	0.00	15.50	0.00

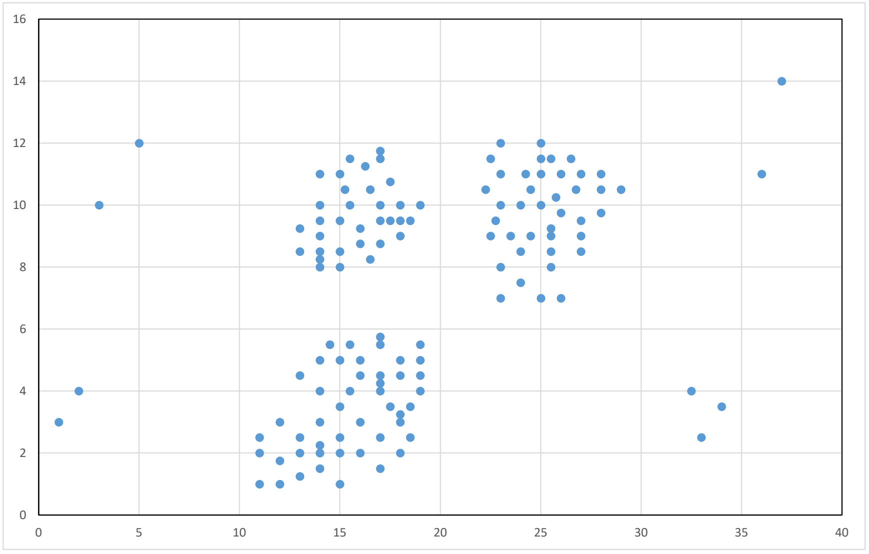


Figure 6. Data set with outliers.

L_∞-norms is robust with respect to outliers, that is it is able to correctly detect and separate outliers.

8. Conclusions.

In this paper, we developed an algorithm for solving the clustering problems. This algorithm can use different similarity measures including those based on the L_1 and L_∞ norms. The algorithm involves a special procedure to generate starting cluster centers for solving both the clustering and auxiliary clustering problems. The discrete gradient method is applied to solve these problems starting from those points. The algorithm was tested using 12 real world data sets. Results demonstrate the ability of the algorithm to find global or near global solutions to the clustering problem.

Results also indicate some limitations of the proposed algorithm. Since the algorithm involves solving of many optimization problems and the size of some of these problems increases as the number of clusters increase the algorithm can become time consuming in very large data sets (with hundreds of thousands or millions of data points). Furthermore, the algorithm may require a large CPU time for solving clustering problems with the large number of clusters (more than 50) in large data sets and also in data sets with large number of attributes.

This algorithm can be applied to the clustering problems with the similarity measure based on any L_p-norm. However, in this case the cluster function is only piecewise partially separable and the discrete gradient method requires more cluster function evaluations than in the case of the L_1 and L_∞ norms. This means that the algorithm may have more limitations on the size of data sets for the general L_p-norms.

In the proposed algorithm we apply local search methods to solve both the clustering and auxiliary clustering problems and use many starting points to find global or near global solutions to clustering problems. Deterministic global optimization methods (for example, those from [28, 34]) can be applied instead of local search methods using one or no starting points. It is interesting to carry out a comparative assessment of this type of algorithms with those introduced in this paper. This will be a subject of the future work.

Acknowledgements. The authors would like to thank two anonymous referees for their comments that helped to improve the quality of the paper. This research by Dr. A. Bagirov was funded by the Australian Government through the Australian Research Council's Discovery Projects scheme (Project No.: DP190100580). The paper was completed when Dr. A. Bagirov visited Chongqing Normal University, China.

REFERENCES

[1] C. Aggarwal, A. Hinneburg and D. Keim, On the surprising behavior of distance metrics in high dimensional space, *ICDT '01 Proceedings of the 8th International Conf. on Database Theory*, (2001), 420–434.

[2] D. Arthur and S. Vassilvitskii, k-means++: The advantages of careful seeding, *SODA '07 Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, (2007), 1027–1035.

[3] K. Bache and M. Lichman, *UCI Machine Learning Repository*, University of California, Irvine, School of Information and Computer Sciences, 2013. Available from: `http://archive.ics.uci.edu/ml`.

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

[5] A. M. Bagirov, A. Al Nuaimat and N. Sultanova, Hyperbolic smoothing function method for minimax problems, *Optimization*, **62** (2013), 759–782.

[6] A. M. Bagirov, B. Karasözen and M. Sezer, Discrete gradient method: Derivative-free method for nonsmooth optimization, *Journal of Optimization Theory and Applications*, **137** (2008), 317–334.

[7] A. M. Bagirov, N. Karmitsa and M. M. Mäkelä, *Introduction to Nonsmooth Optimization*, Springer, Cham, 2014.

[8] A. M. Bagirov and E. Mohebi, An algorithm for clustering using L_1-norm based on hyperbolic smoothing technique, *Computational Intelligence*, **32** (2016), 439–457.

[9] A. M. Bagirov and E. Mohebi, Nonsmooth optimization based algorithms in cluster analysis, *Partitional Clustering Algorithms*, (2015), 99–146.

[10] A. M. Bagirov, B. Ordin, G. Ozturk and A. E. Xavier, An incremental clustering algorithm based on hyperbolic smoothing, *Computational Optimization and Applications*, **61** (2015), 219–241.

[11] A. M. Bagirov, A. M. Rubinov and J. Yearwood, A global optimization approach to classification, *Optimization and Engineering*, **3** (2002), 129–155.

[12] A. M. Bagirov, A. M. Rubinov, N. V. Soukhoroukova and J. Yearwood, Unsupervised and supervised data classification via nonsmooth and global optimization, *TOP*, **11** (2003), 1–93.

[13] A. M. Bagirov and J. Ugon, Piecewise partially separable functions and a derivative-free algorithm for large scale nonsmooth optimization, *Journal of Global Optimization*, **35** (2006), 163–195.

[14] A. M. Bagirov, J. Ugon and D. Webb, Fast modified global k-means algorithm for incremental cluster construction, *Pattern Recognition*, **44** (2011), 866–876.

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

[16] L. Bobrowski and J. C. Bezdek, c-means clustering with the l_1 and l_∞ norms, *IEEE Trans. on Systems, Man and Cybernetics*, **21** (1991), 545–554.

[17] H.-H. Bock, Clustering and neural networks, in *Advances in Data Science and Classification* (eds. A. Rizzi, M. Vichi and H.-H. Bock), Springer, Berlin, (1998), 265–277.

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

[19] M. E. Celebi, H. A. Kingravi and P. A. Vela, A comparative study of efficient initialization methods for the k-means clustering algorithm, *Expert Systems with Applications*, **40** (2013), 200–210.

[20] F. H. Clarke, *Optimization and Nonsmooth Analysis*, Canadian Mathematical Society Series of Monographs and Advanced Texts, Wiley, 1983.

[21] K. A. J. Doherty, R. G. Adams and N. Davey, Non-Euclidean norms and data normalisation, *Proceedings of ESANN*, (2004), 181–186.

[22] M. Ghorbani, Maximum entropy-based fuzzy clustering by using L_1-norm space, *Turk. J. Math.*, **29** (2005), 431–438.

[23] C. Hanilci and F. Ertas, Comparison of the impact of some Minkowski metrics on VQ/GMM based speaker recognition, *Computers and Electrical Engineering*, **37** (2011), 41–56.

[24] P. Hansen, N. Mladenovic and D. Perez-Britos, Variable neighborhood decomposition search, *Journal of Heuristics*, **7** (2001), 335–350.

[25] A. K. Jain, Data clustering: 50 years beyond k-means, *Pattern Recognition Letters*, **31** (2010), 651–666.

[26] A. K. Jain, M. N. Murty and P. J. Flynn, Data clustering: A review, *ACM Comput. Surv.*, **31** (1999), 264–323.

[27] K. Jajuga, A clustering method based on the L_1-norm, *Computational Statistics & Data Analysis*, **5** (1987), 357–371.

[28] D. R. Jones, C. D. Perttunen and B. E. Stuckman, Lipschitzian optimization without the Lipschitz constant, *Journal of Optimization Theory and Applications*, **79** (1993), 157–181.

[29] L. Kaufman and P. J. Rousseeuw, *Finding Groups in Data: An Introduction to Cluster Analysis*, Wiley Series in Probability and Statistics, Wiley, 1990.

[30] J. Kogan, *Introduction to Clustering Large and High-Dimensional Data*, Cambridge University Press, 2007.

[31] A. Likas, N. Vlassis and J. Verbeek, The global k-means clustering algorithm, *Pattern Recognition*, **36** (2003), 451–461.

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

[33] D. Pelleg and A. W. Moore, X-means: Extending k-means with efficient estimation of the number of clusters, *ICML'00 Proceedings of the 17-th International Conference on Machine Learning*, (2000), 727–734.

[34] J. D. Pinter, *Global Optimization in Action. Continous and Lipschitz Optimization: Algorithms, Implementations ans Applications*, Kluwer Academic Publishers, Boston, 1996.

[35] G. N. Ramos, Y. Hatakeyama, F. Dong and K. Hirota, Hyperbox clustering with Ant Colony Optimization method and its application to medical risk profile recognition, *Applied Soft Computing*, **9** (2009), 632–640.

[36] G. Reinelt, TSPLIB - a traveling salesman problem library, *ORSA Journal of Computing*, **3** (1991), 376–384.

[37] K. Sabo, R. Scitovski and I. Vazler, One-dimensional center-based l_1-clustering method, *Optimization Letters*, **7** (2013), 5–22.

[38] R. Sedgewick and K. Wayne, *Introduction to Programming in Java*, Addison-Wesley, 2007.

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

[40] H. Späth, Algorithm 30: L_1 cluster analysis, *Computing*, **16** (1976), 379–387.

[41] N. Venkateswarlu and P. Raju, Fast isodata clustering algorithms, *Pattern Recognition*, **25** (1992), 335–342.

[42] A. E. Xavier and A. A. F. D. Oliveira, Optimal covering of plane domains by circles via hyperbolic smoothing, *J. of Glob. Opt.*, **31** (2005), 493–504.

[43] S. Xu, Smoothing method for minimax problems, *Computational Optimization and Applications*, **20** (2001), 267–279.

[44] R. Xu and D. Wunsch, Survey of clustering algorithms, *IEEE Transactions on Neural Networks*, **16** (2005), 645–678.

[45] M. S. Yang and W. L. Hung, Alternative fuzzy clustering algorithms with L_1-norm and covariance matrix, *Advanced Concepts for Intelligent Vision*, **4179** (2006), 654–665.

[46] J. Zhang, L. Peng, X. Zhao and E. E. Kuruoglu, Robust data clustering by learning multi-metric L_q-norm distances, *Expert Systems with Applications*, **39** (2012), 335–349.

Received August 2018; revised March 2019.

E-mail address: burak.ordin@ege.edu.tr
E-mail address: a.bagirov@federation.edu.au
E-mail address: eh.mohebi@gmail.com