Application of the DIRECT algorithm to searching for an optimal k-partition of the set $\mathcal{A} \subset \mathbb{R}^n$ and its application to the multiple circle detection problem

Rudolf Scitovski[1]
Department of Mathematics, University of Osijek
Trg Ljudevita Gaja 6, HR – 31 000 Osijek, Croatia
e-mail: scitowsk@mathos.hr

Kristian Sabo
Department of Mathematics, University of Osijek
Trg Ljudevita Gaja 6, HR – 31 000 Osijek, Croatia
e-mail: ksabo@mathos.hr

Abstract. In this paper, we propose an efficient method for searching for a globally optimal k-partition of the set $\mathcal{A} \subset \mathbb{R}^n$. Due to the property of the DIRECT global optimization algorithm to usually quickly arrive close to a point of global minimum, after which it slowly attains the desired accuracy, the proposed method uses the well-known k-means algorithm with a initial approximation chosen on the basis of only a few iterations of the DIRECT algorithm. In case of searching for an optimal k-partition of spherical clusters, the method is not worse than other known methods, but in case of solving the multiple circle detection problem, the proposed method shows remarkable superiority.

Key words: globally optimal partition; k-means; Incremental algorithm; DIRECT; multiple circles detection problem;

MSC2010: 65K05, 90C26, 90C27, 90C56, 90C57, 05E05

1 Introduction

A hard partition of the set $\mathcal{A} = \{a^i \in \mathbb{R}^n \colon i = 1, \dots, m\}$ into k nonempty disjoint subsets $\pi_1, \dots, \pi_k$, $1 \le k \le m$ will be denoted by $\Pi(\mathcal{A}) = \{\pi_1, \dots, \pi_k\}$ and the set of all such partitions will be denoted by $\mathcal{P}(\mathcal{A}; k)$. The elements $\pi_1, \dots, \pi_k$ of the partition Π are called *clusters*.

If $d \colon \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}_+$, $\mathbb{R}_+ = [0, +\infty\rangle$ is some distance-like function (see e.g. [16]), then to each cluster $\pi_j \in \Pi$ we can associate its center c_j defined by

$$c_j := \operatorname*{argmin}_{x \in \operatorname{conv}(\mathcal{A})} \sum_{a^i \in \pi_j} d(x, a^i). \qquad (1)$$

[1]Corresponding author: Rudolf Scitovski, e-mail: scitowsk@mathos.hr, telephone number: ++385-31-224-800, fax number: ++385-31-224-801

After that, by introducing the objective function $\mathcal{F}\colon \mathcal{P}(\mathcal{A};k) \to \mathbb{R}_+$, the quality of a partition can be defined, and searching for a *globally optimal k-partition* comes down to solving the following optimization problem:

$$\operatorname*{argmin}_{\Pi\in\mathcal{P}(\mathcal{A};k)} \mathcal{F}(\Pi), \qquad \mathcal{F}(\Pi) = \sum_{j=1}^{k} \sum_{a^i\in\pi_j} d(c_j, a^i), \qquad c = (c_1, \ldots, c_k). \tag{2}$$

Conversely, for a given set of points $c_1, \ldots, c_k \in \mathbb{R}^n$, by applying the minimal distance principle, we can define the partition $\Pi = \{\pi(c_1), \ldots, \pi(c_k)\}$ of the set $\mathcal{A}$ consisting of clusters

$$\pi(c_j) = \{a \in \mathcal{A} : d(c_j, a) \le d(c_s, a),\ \forall s = 1, \ldots, k\}, \qquad j = 1, \ldots, k.$$

Hence, the problem of finding an optimal partition of the set $\mathcal{A}$ can be reduced to the following *global optimization problem* (`GOP`) (see e.g. [16, 35]):

$$\operatorname*{argmin}_{c\in\operatorname{conv}(\mathcal{A})^k} F(c), \qquad F(c) = \sum_{i=1}^{m} \min_{1\le j\le k} d(c_j, a^i). \tag{3}$$

where $F\colon \mathbb{R}^{nk} \to \mathbb{R}_+$. The solutions of (2) and (3) coincide [33, 35].

In this paper, we will use the Least Squares (LS) distance-like function $d(x, y) = \|x - y\|_2^2$.

Clustering a data set into several clusters has a very wide range of applications in multiple areas such as seismic zoning investigation [21, 33], pattern recognition [8, 10], facility location problem, text classification, machine learning, business, biology, agriculture, medicine, psychology, etc. (see e.g. [5, 6, 29]).

For the set of data points $\mathcal{A} \subset \mathbb{R}^n$ with n features, in this paper we propose a method that gives a k-partition near the globally optimal one. The method is based on the k-means algorithm in which the initial approximation has been chosen by using the `DIRECT` global optimization algorithm [14, 15] in a few iterations. For instance, in the test-examples mentioned in Section 2.3, $5-6$ iterations proved to be sufficient. After that, by applying the k-means algorithm, we obtain a partition very close to the globally optimal one very quickly.

The proposed method is also very successfully applied to solving the multiple circle detection problem (see Section 3).

The method was tested on artificial data sets originating from a known partition, which made it possible to check the results by using the adjusted Rand (AR) index [13] and the Hausdorff distance [31].

The results show that the proposed method is not worse in case of searching for an optimal partition consisting of ordinary spherical clusters in $\mathbb{R}^n$, but when it comes to its application to solving the multiple circle detection problem, the proposed method shows remarkable superiority in relation to other test methods. Very short `CPU`-time in this case indicates the possibility of applying it to real-time applications.

The paper is organized as follows. In the next section, we consider the proposed method in case of searching for an optimal k-partition of spherical clusters in $\mathbb{R}^n$. The method is compared with other known methods and tested on 100 randomly generated data sets from $\mathbb{R}^2$ and 100 randomly generated data sets from $\mathbb{R}^5$. In Section 3, we consider an application of the proposed method to solving the multiple circle detection problem. Two variants of the method are presented, which are then compared with other known methods and tested on 100 randomly generated data sets. Finally, some conclusions are discussed in Section 4.

2 Searching for a solution of `GOP` (3)

Given is a set of data points $\mathcal{A} = \{a^i = (a_1^i, \ldots, a_n^i) \in [\alpha, \beta] \colon i = 1, \ldots, m\} \subset \mathbb{R}^n$, where $[\alpha, \beta] = \{x \in \mathbb{R}^n \colon \alpha_i \le x_i \le \beta_i\}$ and $\alpha = (\alpha_1, \ldots, \alpha_n)^T$, $\beta = (\beta_1, \ldots, \beta_n)^T \in \mathbb{R}^n$. The `GOP` (3) is a complex global optimization problem because the objective function $F \colon \mathbb{R}^{nk} \to \mathbb{R}_+$, given by (3), can have a great number of independent variables, it does not have to be either convex or differentiable and generally it may have several local minima, but, as will be shown in the following theorem, the function F is a Lipschitz continuous function.

Theorem 1. *Let $\mathcal{A} = \{a^i \in \mathbb{R}^n \colon i = 1, \ldots, m\} \subset [\alpha, \beta]$ be a set of data points. The function $F \colon [\alpha, \beta]^k \to \mathbb{R}_+$,*

$$F(c) = \sum_{i=1}^{m} \min_{j=1,\ldots,k} \|c_j - a^i\|^2,$$

is a Lipschitz continuous on $[\alpha, \beta]^k$.

Proof. If we define the auxiliary function $F_\varepsilon : [\alpha, \beta]^k \to \mathbb{R}_+$ by

$$F_\varepsilon(u) = -\varepsilon \sum_{i=1}^{m} \log \sum_{j=1}^{k} \exp\Big(-\tfrac{\|c_j - a^i\|^2}{\varepsilon}\Big),$$

then, according to [16], we have

$$0 \le F(u) - F_\varepsilon(u) \le \varepsilon m \log k,$$

and, consequently,

$$\begin{aligned} |F(u) - F(v)| &= |(F(u) - F_\varepsilon(u)) + (F_\varepsilon(v) - F(v)) + (F_\varepsilon(u) - F_\varepsilon(v))| \\ &\le |F(u) - F_\varepsilon(u)| + |F_\varepsilon(v) - F(v)| + |F_\varepsilon(u) - F_\varepsilon(v)| \\ &\le 2\varepsilon m \log k + |F_\varepsilon(u) - F_\varepsilon(v)|. \end{aligned} \tag{4}$$

Because

$$\frac{\partial F_\varepsilon(x)}{\partial x_p} = 2 \sum_{i=1}^{m} \frac{(x_p - a^i) \exp\Big(-\frac{\|x_p - a^i\|^2}{\varepsilon}\Big)}{\sum_{j=1}^{k} \exp\Big(-\frac{\|x_j - a^i\|^2}{\varepsilon}\Big)},$$

it follows

$$\left\|\tfrac{\partial F_\varepsilon(x)}{\partial x_p}\right\| \le 2\sum_{i=1}^{m}\|x_p - a^i\| \le 2\sum_{i=1}^{m}\max_{j=1,\dots,m}\|a^i - a^j\|$$
$$\le 2\,m \max_{i,j\in\{1,\dots,m\}}\|a^i - a^j\|,\ p = 1,\dots,k,$$

i.e. the gradient $\nabla F_\varepsilon(x)$ is continuous and bounded on $[\alpha,\beta]^k$. By using Lagrange's mean value theorem for the function F_ϵ on $[\alpha,\beta]^k$, there exists $L > 0$ (not depending on ε) such that

$$|F_\varepsilon(u) - F_\varepsilon(v)| \le L\|u - v\|, \quad u, v \in [\alpha,\beta]^k.$$

Finally, if $\varepsilon \to 0^+$, from (4) it follows that $|F(u) - F(v)| \le L\|u - v\|$. □

In order to solve GOP (3), we can apply one of the known global optimization methods [12, 17, 20, 34, 39]. For example, the DIRECT optimization algorithm [14, 15], can be applied, but due to a large number of independent variables of the objective function F and the property of the DIRECT algorithm to search for all points of the global minimum, that would be a very inefficient procedure (see Sections 2.3 and 3.4). Namely, in case of searching for an optimal k-partition of the set $\mathcal{A} \subset \mathbb{R}^n$, this means that the algorithm finds at least $k!$ different points in which the global minimum is attained (see [11]).

Some known methods for solving GOP (3), such as different variants of the k-means algorithm [3, 4, 16, 18, 27] or different incremental algorithms [3, 4, 21, 33], give either stationary points or a locally optimal partition, which is highly dependent on the choice of the initial approximation.

In particular, when it comes to data that have only one feature, i.e. $\mathcal{A} \subset \mathbb{R}$, we can apply special global optimization methods for the symmetric Lipschitz continuous function: DISIMPL, SymDIRECT, SepDIRECT (see [11, 23–26, 30]), which give a globally optimal partition. Generally, if $\mathcal{A} \subset \mathbb{R}^n$, $(n > 1)$, the function F given by (3) is a symmetric function in the vectors $c_1,\dots,c_k \in \mathbb{R}^n$ because $F(c_1',\dots,c_k') = F(c_1,\dots,c_k)$ where $(c_1',\dots,c_k')$ is a permutation of the vectors $c_1,\dots,c_k$, but the function F is not symmetric in all its variables and, therefore, the mentioned methods cannot be used in case of $n > 1$.

2.1 GOPart: a new method for solving GOP (3)

In this subsection we will describe a new method for solving GOP (3): *Globally Optimal Partition* (GOPart) method.

Having in mind that DIRECT algorithm arrives close to a global minimum very fast, after which it slowly increases the accuracy (see e.g. [23]), in order to find a solution to GOP (3), the DIRECT algorithm will be used only for achieving a favorable initial approximation for the k-means algorithm. For that particular purpose, the functional $F\colon [\alpha,\beta]^k \to \mathbb{R}$

given by (3) will be transformed into $f\colon [0,1]^{kn} \to \mathbb{R}$, $f(x) = (F \circ T^{-1})(x)$, where the mapping $T\colon [\alpha,\beta]^k \to [0,1]^{kn}$ is given by

$$\begin{aligned} &T(x) = D(x-u), \qquad (5)\\ &D = \operatorname{diag}\left(\tfrac{1}{\beta_1-\alpha_1},\dots,\tfrac{1}{\beta_n-\alpha_n},\dots,\tfrac{1}{\beta_1-\alpha_1},\dots,\tfrac{1}{\beta_n-\alpha_n}\right) \in \mathbb{R}^{(kn)\times(kn)},\\ &u = (\alpha_1,\dots,\alpha_n,\dots,\alpha_1,\dots,\alpha_n) \in \mathbb{R}^{kn}, \end{aligned}$$

and the mapping $T^{-1}\colon [0,1]^{kn} \to [\alpha,\beta]^k$ is given by $T^{-1}(x) = D^{-1}x + u$.

By this transformation GOP (3) becomes the following GOP:

$$\operatorname*{argmin}_{x\in[0,1]^{kn}} f(x), \qquad f(x) = (F\circ T^{-1})(x). \qquad (6)$$

If $\hat{x} \in [0,1]^{kn}$ is an approximation of the solution to GOP (6), then the approximation of the solution to GOP (3) becomes $\hat{c} = T^{-1}(\hat{x})$, where $F(\hat{c}) = F(T^{-1}(\hat{x})) = f(\hat{x})$.

In order to search for a good initial approximation of GOP (6), we will apply the DIRECT algorithm and stop it after a few iterations. In the numerical experiments given below, it was enough to perform only $5-6$ iterations of the DIRECT algorithm.

To the initial approximation obtained in this way, we applied the standard k-means algorithm, which quickly led to a solution very close to the globally optimal one. The corresponding pseudocode for the described method is given in Algorithm 1. Numerous examples presented in Sections 2.3 and 3.4 show the efficiency of the proposed method and this algorithm.

Algorithm 1 : GOPart($\mathcal{A}, k$)

```
Input: $\mathcal{A} \subset [\alpha,\beta]^n$ {Set of data points};  $k \ge 2$  $\epsilon > 0$;
 1: Define the mapping $T^{-1}\colon [0,1]^{kn} \to [\alpha,\beta]^k$, $T^{-1}(x) = D^{-1}x + u$ and the objective
    function $f = F \circ T^{-1}$, where $T$ is given by (5) and $F$ is given by (3);
 2: By using the DIRECT algorithm find the initial approximation $\hat{x} = (\hat{x}_1,\dots,\hat{x}_k) \in$
    $[0,1]^{kn}$ of GOP (6);
 3: By using the standard $k$-means algorithm with the initial approximation $\hat{x}$ determine
    cluster centers $x^\star = (x_1^\star,\dots,x_k^\star)$;
 4: Calculate $c^\star = (c_1^\star,\dots,c_k^\star) = T^{-1}(x^\star) \in [\alpha,\beta]^k$;
Output: $\{c^\star, F(x^\star)\}$.
```

2.2 Comparison with some known methods

The efficiency of the proposed GOPart method and corresponding algorithm will be compared with some known frequently cited algorithms, such as the DIRECT algorithm, the Multistart k-means algorithm, and the Incremental algorithm.

2.2.1 The DIRECT algorithm

A derivative-free, deterministic sampling method for global optimization of a Lipschitz continuous function $g\colon \mathcal{D} \to \mathbb{R}$ defined on a bound-constrained region $\mathcal{D} \subset \mathbb{R}^p$ named

Dividing Rectangles (DIRECT) was proposed by [15]. The function g is first transformed into $f\colon [0,1]^p \to \mathbb{R}$, and after that, by means of a standard strategy (see, e.g. [9, 14, 15]), the unit hypercube $[0,1]^p$ is divided into smaller hyperrectangles, among which the so-called potentially optimal ones are first searched for and then further divided. It should be noted that this procedure does not assume knowing the Lipschitz constant $L > 0$.

Searching for a globally optimal partition by using the DIRECT algorithm has proved to be insufficiently efficient (see numerical experiments in Section 2.3 and 3.4). Namely, as mentioned earlier (see e.g. [23]), the DIRECT algorithm quickly arrives close to a point of global minimum, but it can be very slow when it needs to attain high accuracy. Apart from that, in our case (6) the set $\operatorname*{argmin}_{x\in[0,1]^{kn}} f(x)$ contains at least $k!$ different points of global minimum (see also [11, 30]), and DIRECT algorithm will search through all those points.

2.2.2 The Multistart k-means algorithm

GOPart algorithm will also be compared with the k-means algorithm [16, 27, 32], where the initial centers are chosen in many successive iterations and a better solution is retained [18]. For more details about global optimality in the k-means algorithm, see [37]. This procedure is written in Algorithm 2.

Algorithm 2 : (Multistart k-means algorithm)

```
Input: $\mathcal{A} \subset [\alpha,\beta] \subset \mathbb{R}^n$ {Set of data points};   $k \geq 2$   $It > 1$;
1: Determine $c^{(0)} \in [\alpha,\beta]^k$ at random;
2: Apply the $k$-means algorithm to the set $\mathcal{A}$, with initial centers $c^{(0)}$, denote the solution
   by $\hat{c} = \hat{c}^{(0)}$ and set $F_0 = F(\hat{c})$;
3: for $i = 1$ to $It$ do
4:     Determine $c^{(i)} \in [\alpha,\beta]^k$ at random;
5:     Apply the $k$-means algorithm to the set $\mathcal{A}$, with initial centers $c^{(i)}$, denote the
       solution by $\hat{c}^{(i)}$ and set $F_1 = F(\hat{c}^{(i)})$;
6:     if $F_1 \leq F_0$ then
7:         Set $\hat{c} = \hat{c}^{(i)}$ and set $F_0 = F_1$;
8:     end if
9: end for
Output: $\{\hat{c}, F(\hat{c})\}$.
```

Remark 1. Note that line 6 includes the possibility that the k-means algorithm loses some cluster. In that case, the value of the function F increases, so that this partition is not competitive in terms of an optimal partition.

The algorithm for circle centers that will be used in Section 3 is defined analogously.

2.2.3 The Incremental algorithm

The Incremental algorithm [3, 4, 33], which emerged as an improvement of the global k-means algorithm originally proposed in [19], is very frequently mentioned in the literature as a tool for searching for a partition close to a globally optimal one. The proposed GOPart algorithm will be compared with an incremental algorithm in the way it was constructed in [33]. After determining the first $r-1$ centers $\hat{c}_1,\ldots,\hat{c}_{r-1}$, an approximation of the following r-th center is determined by using DIRECT algorithm to solve the following GOP:

$$\hat{c}_r = \operatorname*{argmin}_{c\in\mathbb{R}^n} \Phi(c), \quad \Phi(c) = \sum_{i=1}^{m} \min\{\delta_{r-1}^i, d(c,a^i)\}, \tag{7}$$

where $\delta_{r-1}^i = \min\{d(\hat{c}_1,a^i),\ldots,d(\hat{c}_{r-1},a^i)\}$. After that, the first r centers $c_1^\star,\ldots,c_r^\star$ are obtained by using the k-means algorithm with initial centers $\hat{c}_1,\ldots,\hat{c}_r$. The main shortcoming of this algorithm is its strong dependence on the choice of the initial center $\hat{c}_1$. A reasonable possibility for that is the mean of the set $\mathcal{A}$ or random choice [3, 4, 27].

2.3 Numerical experiments

For the purpose of comparing the proposed algorithm with other algorithms listed in the previous section, we carried out the following experiment[2]. In the square $[0,10]^2\subset\mathbb{R}^2$, $k=5$ different points $P_j\in[0,10]^2$, $j=1,\ldots,k$ were randomly chosen, such that $\|P_r-P_s\|>1$ for $r\neq s$. After that, $m_j\sim\mathcal{U}(280,320)$ random points were generated in the neighborhood of the point C_j by using binormal random additive errors with mean vector $\mathbf{0}\in\mathbb{R}$ and the covariance matrices $\sigma_j^2\mathbf{I}$, $\sigma_j^2\in[1,1.25]$, where $\mathbf{I}\in\mathbb{R}^{2\times2}$ is the identity matrix. These points make the cluster π_j. In this way, we construct the partition $\Pi=\{\pi_1,\ldots,\pi_k\}$ of the set $\mathcal{A}$ with clusters π_j and their centers $c_j=\frac{1}{m_j}\sum\limits_{a\in\pi_j}a$.

Additionally, the cluster π_j will be characterized by the circle

$$C_j(c_j,\sigma_j)=\{x\in\mathbb{R}^2\colon \|c_j-x\|=\sigma_j\}, \quad \sigma_j^2=\tfrac{1}{|\pi_j|}\sum_{a\in\pi_j}\|c_j-a\|^2, \tag{8}$$

which will be called the *main circle* of the cluster π_j.

The efficiency of an algorithm will be measured by its ability to recognize the partition Π as well as by CPU-time required for that purpose. That is why we will use this algorithm to determine the optimal partitions $\hat{\Pi}^{(s)}$, $s=2,\ldots,6$, and calculate the corresponding AR index $R(\hat{\Pi}^{(s)},\Pi)$ for each of them (see e.g. [13]). We will consider the partition $\hat{\Pi}^{(s_0)}$ as the best partition of the set $\mathcal{A}$ if the highest AR index is reached thereon.

In addition, the quality of the obtained partition $\hat{\Pi}^{(s_0)}$ will be measured by comparing its main circles with the main circles (8) of the original partition Π. If $\hat{c}=(\hat{c}_1,\ldots,\hat{c}_{s_0})$ are centers, and $\hat{C}_t(\hat{c}_t,\hat{\sigma}_t)$, $\hat{\sigma}_t^2=\frac{1}{|\hat{\pi}_t|}\sum\limits_{a\in\hat{\pi}_t}\|\hat{c}_t-a\|^2$ the corresponding main circles of clusters $\hat{\pi}_1,\ldots,\hat{\pi}_{s_0}$, we will consider that some main circle C_j is recognized if there exists a main circle $\hat{C}_t(\hat{c}_t,\hat{\sigma}_t)$ from the partition $\hat{\Pi}^{(s_0)}$ such that the Hausdorff distance [31]

$$H(C_j,\hat{C}_t)=\|c_j-\hat{c}_t\|+|\sigma_j-\hat{\sigma}_t|<\epsilon, \tag{9}$$

[2]All evaluations were done on the basis of our own *Mathematica*-modules freely available at: `https://www.mathos.unios.hr/images/homepages/scitowsk/GOPart.rar`, and were performed on the computer with a `2.90 GHz Intel(R) Core(TM)i7-75000 CPU` with `16GB` of `RAM`.

for some small $\epsilon > 0$. The recognized main circles were searched for by means of Algorithm 3 (see Section 3.4.1).

The algorithms considered in this way, i.e. the `DIRECT` algorithm, the `Multistart` k-`means` algorithm (with $It = 25$ randomly chosen initial approximations), the `Incremental` algorithm and the new `GOPart` algorithm, will be tested and their efficiency will be mutually compared on 100 data sets randomly generated in the previously described way. Initial approximation for the `GOPart` algorithm has been obtained in 5 iterations of the `DIRECT` algorithm. Table 1 gives realized characteristics of every aforementioned algorithm:

- `CPU`-time required;
- the number of experiments in which the partition with $s \in \{2, \dots, 6\}$ clusters was chosen as the best partition on the basis of the AR index (see the first multicolumn in Table 1);
- the number of the main circles in the best partition recognized by using Algorithm 3 with threshold $\epsilon = .25$ (see the second multicolumn in Table 1).

In Table 1, it can be seen that the proposed `GOPart` algorithm is not worse than other algorithms, and the `DIRECT` algorithm requires significantly more `CPU`-time than other algorithms.

Algorithm	`CPU`-time	Detection of the best partition					Main circles recognized					
	(sec.)	6	5	4	3	2	5	4	3	2	1	0
`GOPart`	12.80	2	91	7	-	-	18	26	28	18	9	1
`Incremental`	8.41	3	88	9	-	-	15	28	18	25	11	3
k-`means`	98.13	-	92	8	-	-	18	17	33	19	8	5
`DIRECT`	688.21	-	100	-	-	-	21	31	25	23	-	-

Table 1: Frequency of detection of the best partition with $s \in \{2,3,4,5,6\}$ clusters, frequency of the main circles recognized and `CPU`-time required by algorithms for the set $\mathcal{A} \subset \mathbb{R}^2$

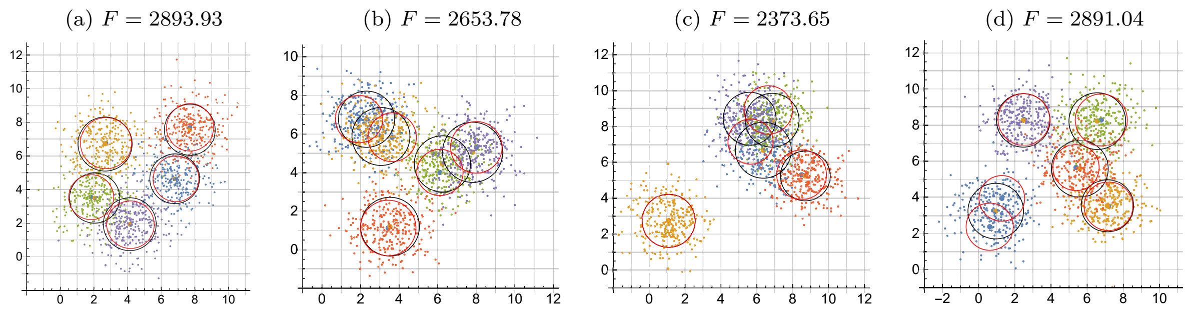

Figure 1: Examples of partitions from the described experiment

Fig.1 shows four selected data sets with corresponding cluster centers and their main circles (black circles), and objective function values are given in the header. Figures 1a-b show sets of data points, where the GOPart algorithm recognized the 5-partition as the best partition. All five main circles (red circles) were recognized in Fig.1a, whereas only one main circle was recognized in Fig.1b. For the set of data points shown in Fig.1c, the GOPart algorithm recognized the 4-partition as the best partition and only two main circles. For the set of data points shown in Fig.1d, the GOPart algorithm recognized the 6-partition as the best partition and only two main circles.

The same experiment was also conducted on 100 similarly generated sets $\mathcal{A} \subset \mathbb{R}^5$. Initial approximation for the proposed GOPart algorithm has been obtained in 6 iterations of the DIRECT algorithm. It was also shown that the proposed GOPart algorithm is not worse than other algorithms.

Algorithm	CPU-time (sec.)	Detection of the best partition 6	5	4	3	2	Main circles recognized 5	4	3	2	1	0
GOPart	22.32	4	94	2	-	-	82	2	6	10	-	-
Incremental	129.27	3	91	6	-	-	80	9	6	5	-	-
k-means	114.13	3	92	5	-	-	90	-	5	3	1	1
DIRECT	1490.83	-	100	-	-	-	81	-	12	7	-	-

Table 2: Frequency of detection of the best partition with $s \in \{2,3,4,5,6\}$ clusters, frequency of the main circles recognized and CPU-time required by algorithms for the set $\mathcal{A} \subset \mathbb{R}^5$

The Multistart k-means algorithm was run on the basis of 25 random initial approximations and significant CPU-time was necessary for running this algorithm. The Incremental algorithm also shows relatively good results.

The proposed GOPart algorithm has a very high degree of recognition and small CPU-time justified the initial expectations. It should also be noted that for the implementation of the GOPart algorithm, in Step 2 the DIRECT algorithm required an average of one-third, whereas in Step 3 the k-means algorithm required two-thirds of the total CPU-time.

These simple illustrative examples show that the characteristics of the proposed GOPart algorithm are not worse than other algorithms compared. Its superiority, when it comes to solving the multiple circles detection problem, will be shown in the next section.

3 Application to solving the multiple circle detection problem

Let $\mathcal{A} = \{a^i = (x_i, y_i) \in \mathbb{R}^2 \colon \alpha_1 \leq x_i \leq \beta_1, \alpha_2 \leq y_i \leq \beta_2, i = 1, \ldots, m\}$ be a set of points which come from k circles that should be reconstructed or detected. Note that $\mathcal{A} \subset [\alpha, \beta] = [\alpha_1, \beta_1] \times [\alpha_2, \beta_2] \subset \mathbb{R}^2$, $\alpha = (\alpha_1, \alpha_2)$, $\beta = (\beta_1, \beta_2)$. There are several different approaches to solving this problem in the literature, such as methods based on Hough transformation and various heuristic approaches (see e.g. [2, 8, 28]). Most of them cannot be used in real-time applications.

In [31], this problem is considered as a center-based clustering problem, where centers of clusters are circles. Based on this method, in our paper we propose a new, very efficient method for solving this problem. We shall name this method *Multiple Circle Detection* (MCD) method.

Searching for an optimal partition $\Pi^\star = \{\pi_1^\star, \dots, \pi_k^\star\}$ with cluster circle-centers $C_j^\star(S_j^\star, r_j^\star)$, $S_j^\star = (p_j^\star, q_j^\star)$ boils down to searching for optimal parameters $(p_j^\star, q_j^\star, r_j^\star)$, $j = 1, \dots, k$, which give a solution to the following GOP (cf. (3))

$$\operatorname*{argmin}_{(\mathbf{p},\mathbf{q})\in[\alpha,\beta]^k, \mathbf{r}\in[0,R]^k} F(\mathbf{p},\mathbf{q},\mathbf{r}), \quad F(\mathbf{p},\mathbf{q},\mathbf{r}) = \sum_{i=1}^{m} \min_{1\le j\le k} \{D((p_j,q_j),r_j),a^i)\}, \tag{10}$$

where $\mathbf{p} \in [\alpha_1, \beta_1]^k$, $\mathbf{q} \in [\alpha_2, \beta_2]^k$, $\mathbf{r} \in [0, R]^k$, $R = \frac{1}{2}\min\{\alpha_2 - \alpha_1, \beta_2 - \alpha_2\}$, and $D((p_j,q_j),r_j),a^i)$ represent the distance from the point $a^i \in \mathcal{A}$ to the circle $C_j(p_j,q_j)$. The distance-like function D can be defined in a different way [7, 22, 31], but the *algebraic distance*

$$D(C(S,r),a^i) = (\|S - a^i\|^2 - r^2)^2 \tag{11}$$

occurs most frequently in applications, and, therefore, this possibility is also used in our paper.

3.1 MCD method

In line with the GOPart method described in Section 2.1, it is possible to apply the global optimization algorithm DIRECT for the purpose of finding a favorable initial approximation. For that purpose, similarly to Section 2.1, the objective function $F\colon [\alpha,\beta]^k \times [0,R]^k \to \mathbb{R}$ will be transformed on $f\colon [0,1]^{3k} \to \mathbb{R}$, $f(x) = (F \circ T^{-1})(x)$, where the mapping $T\colon [\alpha_1,\beta_1]^k \times [\alpha_2,\beta_2]^k \times [0,R]^k \to [0,1]^{3k}$ is given by

$$\begin{aligned} & T(x) = D(x - u), \\ & D = \operatorname{diag}\left(\tfrac{1}{\beta_1-\alpha_1}, \tfrac{1}{\beta_2-\alpha_2}, \tfrac{1}{R}, \dots \tfrac{1}{\beta_1-\alpha_1}, \tfrac{1}{\beta_2-\alpha_2}, \tfrac{1}{R}\right) \in \mathbb{R}^{3k\times 3k} \\ & u = (\alpha_1, \beta_1, 0, \dots, \alpha_1, \beta_1, 0) \in \mathbb{R}^{3k}. \end{aligned} \tag{12}$$

An initial approximation $\hat{\mathbf{x}} \in [0,1]^{3k}$ for the GOP

$$\operatorname*{argmin}_{x\in[0,1]^{3k}} f(x), \quad f(x) = (f \circ T^{-1})(x), \tag{13}$$

will be determined by using the DIRECT algorithm. The vector $(\hat{\mathbf{p}}, \hat{\mathbf{q}}, \hat{\mathbf{r}}) = T^{-1}(\hat{\mathbf{x}})$ is an initial approximation for solving GOP (10). After that, a globally optimal solution of (10) will be obtained by applying the k-closest circles algorithm (KCC). This algorithm is the well-known k-means algorithm [16, 18] adapted for searching for a locally optimal partition with circles as clusters-centers (see [31]). The algorithm can be described in two steps which are repeated iteratively.

Algorithm 3. (The k-closest circles algorithm (`KCC`))

Step A: For each set of mutually different circles $C_1,\dots,C_k$, the set $\mathcal{A}$ should be divided into k disjoint unempty clusters $\pi_1,\dots,\pi_k$ by using the minimal distance principle:

$$\pi_j := \pi_j(C_j) = \{a \in \mathcal{A}\colon D(C_j,a) \le D(C_s,a),\ \forall s = 1,\dots,k,\ s \ne j\}; \tag{14}$$

Step B: Given a partition $\Pi = \{\pi_1,\dots,\pi_k\}$ of the set $\mathcal{A}$, one can define the corresponding circle-centers $C_j^\star((p_j^\star,q_j^\star),r_j^\star)$ $j = 1,\dots,k$ by solving the following `GOP`s

$$\operatorname*{argmin}_{(p,q)\in[\alpha,\beta],r\in[0,R]} F_j(p,q,r), \quad F_j(p,q,r) = \sum_{a\in\pi_j} D(C((p,q),r),a); \tag{15}$$

Remark 2. Solutions to `GOP`s (15) can be found by using some local optimization method (Newton, Quasi-Newton), since for every $j = 1,\dots,k$ in the cluster π_j we are able to determine a very favorable initial approximation $\hat{C}_j(\hat{S}_j,\hat{r}_j)$ of the required circle. Namely, for $\hat{S}_j$, we can choose a centroid $\frac{1}{|\pi_j|}\sum\limits_{a\in\pi_j} a$ of the cluster π_j, and $\hat{r}_j$ is determined by

$$\hat{r}_j^2 = \tfrac{1}{|\pi_j|} \sum_{a\in\pi_j} \|\hat{S}_j - a\|^2, \tag{16}$$

because

$$\sum_{a\in\pi_j} \left(\|\hat{S}_j - a\|^2 - r_j^2\right)^2 \ge \sum_{a\in\pi_j} \left(\|\hat{S}_j - a\|^2 - \hat{r}_j^2\right)^2, \quad \text{for all } r_j \in \mathbb{R}.$$

After applying the `KCC` algorithm to the vector $(\hat{\mathbf{p}},\hat{\mathbf{q}},\hat{\mathbf{r}})$, we obtain a globally optimal solution $(\mathbf{p}^\star,\mathbf{q}^\star,\mathbf{r}^\star)$ of `GOP` (10).

3.2 Modified `MCD` method (`MMCD`)

In `MCD` algorithm, the initial circle centers are searched for by using the `DIRECT` algorithm. This means that `GOP` (13) is solved after transformation (12), where the objective function f has $3k$ independent variables. It can be seen that this number can be reduced to $2k$, without losing efficiency of the algorithm. Simply, instead of solving `GOP` (10), we are solving

$$\operatorname*{argmin}_{(\mathbf{p},\mathbf{q})\in[\alpha,\beta]^k} \tilde{F}(\mathbf{p},\mathbf{q}), \quad \tilde{F}(\mathbf{p},\mathbf{q}) = \sum_{i=1}^m \min_{1\le j\le k} \{D((p_j,q_j),r_j),a^i)\}, \tag{17}$$

where $r_j \in [0,R]$ are constants (say $r_j = 1$). To the initial approximation obtained in this way, we apply the `KCC` algorithm, which gives an optimal partition.

3.3 Comparison with other algorithms

The proposed `MCD` and `MMCD` algorithms will be compared with the `Multistart` k`-means` algorithm for circle-centers and with the `Incremental` algorithm for circle-centers [31] and with the `DIRECT` algorithm. Akinlar and Topal [2] have proposed a real-time, parameter-free circle detection (`Algorithm EDCircles`) with high detection rates, but this algorithm is not applicable to solving the circle detection problem in the case of circles with unclear or noisy edges and therefore it does not recognize any of the circles tested in Section 3.4.

3.3.1 The Multistart k-means algorithm for circle-centers

The `Multistart` k`-means` algorithm for circle-centers can be constructed similarly to Algorithm 2, where the initial approximation is a vector consisting of k circle-centers $\hat{C}_j(\hat{S}_j, \hat{r}_j)$, $j = 1, \dots, k$, where $\hat{S}_j \in [\alpha, \beta] \subset \mathbb{R}^2$ are randomly selected points such that $\|\hat{S}_r - \hat{S}_s\| > 1$ for $r \neq s$, a $\hat{r}_j \sim \mathcal{U}(0, R)$. Similarly to Algorithm 2, the initial circle-centers are successively selected multiple times, we apply the `KCC` algorithm thereto and retain a better solution.

3.3.2 Incremental algorithm for circle-centers

In [31], the authors proposed a modification of the incremental algorithm for solving the multiple circle detection problem. After determining the first $r-1$ circle-centers $\hat{C}_1, \dots, \hat{C}_{r-1}$, the approximation of the following r-th circle-center $\hat{C}_r$ is determined by solving the following `GOP`

$$\operatorname*{argmin}_{p,q\in[\alpha,\beta],\, r\in[0,R]} \Phi(p,q,r), \qquad \Phi(p,q,r) = \sum_{i=1}^{m} \min\{\delta_{r-1}^i, D(C((p,q),r), a^i)\}, \tag{18}$$

where $\delta_{r-1}^i = \min\{D(\hat{C}_1, a^i), \dots, D(\hat{C}_{r-1}, a^i)\}$. The solution to `GOP` (18) will also be searched for by using the `DIRECT` algorithm. After that, the first r circle-centers $C_1^\star, \dots, C_r^\star$ are obtained by using the `KCC` algorithm with initial circle-centers $\hat{C}_1, \dots, \hat{C}_r$.

The main shortcoming of this algorithm is its dependence on the initial circle-center $\hat{C}_1$. A reasonable possibility for that is $\hat{C}_1 = C(\hat{S}_1, \hat{r}_1)$, where, in line with Remark 2, we choose

$$\hat{S}_1 = \tfrac{1}{m} \sum_{a\in\mathcal{A}} a, \qquad \hat{r}_1 = \sqrt{\tfrac{1}{m} \sum_{a\in\mathcal{A}} \|\hat{S}_1 - a\|^2}.$$

Another possibility is the choice of a random circle-center.

3.4 Numerical experiments

The following experiment was conducted. In the square $[0,10]^2 \subset \mathbb{R}^2$, we randomly choose k points $S_1, \dots, S_k$, such that $\|S_r - S_s\| > 2$ for $r \neq s$ and k random real numbers $r_i \sim \mathcal{U}(.5, 2)$. In this way, we construct a set of circles $\mathcal{C} = \{C_j(S_j, r_j) \colon j = 1, \dots, k\}$ in the plane. In the neighborhood of each circle C_j, $m_j = 300\, r_j$ random points were generated by using binormal random additive errors with mean vector $\mathbf{0} \in \mathbb{R}$ and covariance matrices $\sigma_j^2 \mathbf{I}$, $\sigma_j^2 \in [1, 1.25]$, where $\mathbf{I} \in \mathbb{R}^{2\times 2}$ is the identity matrix. These points make a cluster π_j. The partition $\Pi = \{\pi_1, \dots, \pi_k\}$ and the set of data points $\mathcal{A}$ are constructed in this way. Circle-centers $\tilde{C}_j$ of the cluster π_j are determined by using the `KCC` algorithm with the initial approximation C_j.

Similarly to Section 2.3, efficiency of algorithms under consideration used for searching for the optimal k-partition of the set $\mathcal{A}$ (`MCD, MMCD, Incremental,` k`-means, and DIRECT`) will be measured by its ability to recognize the original circles as well as `CPU`-time required for that purpose.

The initial approximation for the MCD algorithm has been obtained in 10 iterations and, after that, the optimal k-partition has been obtained by using KCC algorithm in $3-18$ iterations. The initial approximation for the MMCD algorithm has been obtained in 15 iterations and, after that, the optimal k-partition has been obteined by using KCC algorithm in $5-21$ iterations. Which circles are recognized and the total number of recognized circles will be determined by using Algorithm 3 given in Section 3.4.1.

In this way, we will test and compare efficiency of algorithms under consideration on 100 sets of data points randomly generated in the previously described way for $k = 5$. Table 3 gives realized characteristics of every aforementioned algorithm, CPU-time required and the number of recognized circles.

Algorithm	CPU (sec.)	No. of recognized circles 5	4	3	2	1	0
MCD	2.47	87	9	4	-	-	-
MMCD	1.96	88	9	2	1	-	-
Incremental	36.85	26	3	25	21	17	8
k-means	38.00	89	4	6	1	-	-
DIRECT	459.34	85	6	9	1	-	-

Table 3: Frequency of the number of recognized circles and CPU-time required by algorithms

As can be seen in Table 3, CPU-time required for performing the proposed MCD and MMCD algorithms is significantly shorter with respect to other test methods, and a degree of recognition is satisfactory. In addition, let us also mention that, for the implementation of the proposed MCD and MMCD algorithms, approximately equivalent CPU-time was necessary for determining the initial approximation (by using the DIRECT algorithm) and for searching for the final solution by using the KCC algorithm.

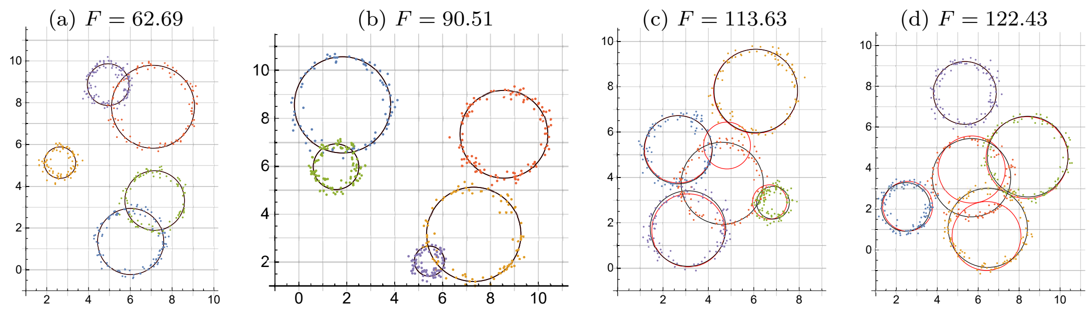

Figure 2: Examples of partitions from described experiments

Fig. 2 shows four selected data sets with corresponding original black circles and calculated red circles for which the process of recognition was performed by using MCD algorithm. Objective function values for original data sets are given in the header. Figures 2a-b show two different sets of data points, where all five circles were recognized. Fig. 2c and Fig. 2d show sets of data points, where four circles and three circles were recognized, respectively.

In case the set of data points $\mathcal{A}$ originates from an unknown number of unknown circles, the proposed algorithm should be applied to searching for optimal partitions with $2, 3 \ldots$ clusters, and the choice of a partition with the most appropriate number of clusters can be done on the basis of the Davies-Bouldin index (see [10, 31]).

3.4.1 Similarity measure for pairs of circles

Similarly to [10, 31], in every experiment it was necessary to establish how many circles were recognized and which circles those were exactly (the second multicolumn in Table 1, Table 2 and Table 3). `Algorithm 3` compares reconstructed circles $\hat{C}_s$, $s = 1, \ldots, r$ with original circles C_j, $j = 1, \ldots, k$ and gives the answer to these questions.

Algorithm 3 (Search for detected circles)

```
Input: $C_j$, $j = 1, \ldots, k$ (Original circles),   $\hat{C}_s$, $s = 1, \ldots, r$ (Calculated circles),   $\epsilon > 0$
 1: for $s = 1, \ldots, r$ do
 2:     $min = 10\epsilon$;
 3:     for $j = 1, \ldots, k$ do
 4:         $H_{dist} = H(\hat{C}_s, C_j)$   [According to (9)]
 5:         if $H_{dist} < min$, then
 6:             $min = H_{dist}$; $j_0 = j$
 7:         end if
 8:     end for
 9:     if $min < \epsilon$,  then
10:         $nr = nr + 1$   “Circle $\hat{C}_s$ is recognized as the circle $C_{j_0}$”
11:     end if
12: end for
Output: $nr$
```

4 Conclusions

Searching for a globally optimal partition of the set $\mathcal{A} \subset \mathbb{R}^n$ by using some global optimization method is not acceptable due to the existence of a large number of points where the global minimum of the corresponding objective function is attained. Hence, in this paper we propose the use of the `DIRECT` global optimization algorithm only for the purpose of searching for a good initial approximation, after which, the standard k-means algorithm should be applied. In terms of efficiency, such approach is not worse than other known methods used for searching for an optimal partition with spherical clusters in $\mathbb{R}^n$.

However, if this approach is used for solving the multiple circle detection problem, the obtained results are much better. It is shown that the proposed algorithm gives a correct solution to this problem and that `CPU`-time required is rather short at the same time

A similar result can also be expected in the case of applications to other geometrical objects (e.g. lines [38], ellipses [10], generalized circles [36]).

Short CPU-time necessary for the implementation of the proposed algorithm indicates a possibility of its application to solving problems in real-time applications.

Acknowledgements.

The author would like to thank the referees and the journal editors for their careful reading of the paper and insightful comments that helped us improve the paper. Especially, the author would like to thank Mrs. Katarina Moržan for significantly improving the use of English in the paper. This work was supported by the Croatian Science Foundation through research grants IP-2016-06-6545 and IP-2016-06-8350.

References

[1] Ahn, S.J., Rauh, W., Warnecke, H.J., 2001. Least-squares orthogonal distances fitting of circle, sphere, ellipse, hyperbola, and parabola. Pattern Recognition 34, 2283–2303.

[2] Akinlar, C., Topal, C., 2013. Edcircles: A real-time circle detector with a false detection control. Pattern Recognition 46, 725–740.

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

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

[5] Bezdek, J.C., Keller, J., Krisnapuram, R., Pal, N.R., 2005. Fuzzy models and algorithms for pattern recognition and image processing. Springer, New York.

[6] Butenko, S., Chaovalitwongse, W.A., Pardalos, P.M. (Eds.), 2009. Clustering Challenges in Biological Networks, World Scientific Publishing Co.

[7] Chernov, N., 2010. Circular and linear regression: Fitting circles and lines by least squares. volume 117 of *Monographs on Statistics and Applied Probability*. Chapman & Hall/CRC, London.

[8] Chung, K.L., Huang, Y.H., Shen, S.M., Yurin, A.S.K.D.V., Semeikina, E.V., 2012. Efficient sampling strategy and refinement strategy for randomized circle detection. Pattern Recognition 45, 252–263.

[9] Gablonsky, J.M., 2001. DIRECT Version 2.0. Technical Report. Center for Research in Scientific Computation. North Carolina State University.

[10] Grbić, R., Grahovac, D., Scitovski, R., 2016. A method for solving the multiple ellipses detection problem. Pattern Recognition 60, 824–834.

[11] Grbić, R., Nyarko, E.K., Scitovski, R., 2013. A modification of the DIRECT method for Lipschitz global optimization for a symmetric function. Journal of Global Optimization 57, 1193–1212.

[12] Horst, R., Tuy, H., 1996. Global Optimization: Deterministic Approach. Springer. 3rd, revised and enlarged edition.

[13] Hüllermeier, E., Rifqi, M., Henzgen, S., Senge., R., 2012. Comparing fuzzy partitions: A generalization of the Rand index and related measures. EEE Transactions on Fuzzy Systems, 20, 546–556.

[14] Jones, D.R., 2001. The direct global optimization algorithm, in: Floudas, C.A., Pardalos, P.M. (Eds.), The Encyclopedia of Optimization. Kluwer Academic Publishers, Dordrect, pp. 431–440.

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

[16] Kogan, J., 2007. Introduction to Clustering Large and High-dimensional Data. Cambridge University Press, New York.

[17] Kvasov, D.E., Sergeyev, Y.D., 2012. Lipschitz gradients for global optimization in a one-point-based partitioning scheme. Journal of Computational and Applied Mathematics 236, 4042 – 4054.

[18] Leisch, F., 2006. A toolbox for k-centroids cluster analysis. Computational Statistics & Data Analysis 51, 526–544.

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

[20] Locatelli, M., Schoen, F., 2013. Global Optimization: Theory, Algorithms, and Applications. SIAM and the Mathematical Optimization Society.

[21] Morales-Esteban, A., Martínez-Álvarez, F., Scitovski, S., Scitovski, R., 2014. A fast partitioning algorithm using adaptive Mahalanobis clustering with application to seismic zoning. Computers & Geosciences 73, 132–141.

[22] Nievergelt, Y., 2002. A finite algorithm to fit geometrically all midrange lines, circles, planes, spheres, hyperplanes, and hyperspheres. Numerische Mathematik 91, 257–303.

[23] Paulavičius, R., Sergeyev, Y., Kvasov, D., Žilinskas, J., 2014. Globally-biased DISIMPL algorithm for expensive global optimization. Journal of Global Optimization 59, 545–567.

[24] Paulavičius, R., Žilinskas, J., 2014a. Simplicial Global Optimization. Springer.

[25] Paulavičius, R., Žilinskas, J., 2014b. Simplicial Lipschitz optimization without Lipschitz constant. Journal of Global Optimization 59, 23–40.

[26] Paulavičius, R., Žilinskas, J., 2016. Advantages of simplicial partitioning for Lipschitz optimization problems with linear constraints. Optimization Letters 10, 237–246.

[27] Pelleg, D., Moore, A.W., 2000. X-means: Extending k-means with efficient estimation of the number of clusters, in: Proceedings of the Seventeenth International Conference on Machine Learning, ICML'00, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. pp. 727–734.

[28] Qiao, Y., Ong, S.H., 2004. Connectivity-based multiple-circle ftting. Pattern Recognition 37, 755–765.

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

[30] Scitovski, R., 2017. A new global optimization method for a symmetric Lipschitz continuous function and application to searching for a globally optimal partition of a one-dimensional set. Journal of Global Optimization 68, 713–727.

[31] Scitovski, R., Marošević, T., 2014. Multiple circle detection based on center-based clustering. Pattern Recognition Letters 52, 9–16.

[32] Scitovski, R., Sabo, K., 2014. Analysis of the k-means algorithm in the case of data points occurring on the border of two or more clusters. Knowledge-Based Systems 57, 1–7.

[33] Scitovski, R., Scitovski, S., 2013. A fast partitioning algorithm and its application to earthquake investigation. Computers & Geosciences 59, 124–131.

[34] Sergeyev, Y.D., Kvasov, D.E., 2015. A deterministic global optimization using smooth diagonal auxiliary functions. Commun Nonlinear Sci Numer Simulat 21, 99–111.

[35] Späth, H., 1983. Cluster-Formation und Analyse. R. Oldenburg Verlag, München.

[36] Thomas, J.C.R., 2011. A new clustering algorithm based on k-means using a line segment as prototype, in: Martin, C.S., Kim, S.W. (Eds.), Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications, Springer Berlin Heidelberg. pp. 638–645.

[37] Tîrnăucă, C., Gómez-Pérez, D., Balcázar, J.L., Montaña, J.L., 2018. Global optimality in k-means clustering. Information Sciences 439, 79–94.

[38] Vidović, I., Scitovski, R., 2014. Center-based clustering for line detection and application to crop rows detection. Computers and Electronics in Agriculture 109, 212–220.

[39] Weise, T., 2008. Global Optimization Algorithms. Theory and Application. e-book: http://www.it-weise.de/projects/book.pdf.