KYBERNETIKA — VOLUME *50* (2014), NUMBER 5, PAGES 661–678

CONSENSUS CLUSTERING WITH DIFFERENTIAL EVOLUTION

Miroslav Sabo

Consensus clustering algorithms are used to improve properties of traditional clustering methods, especially their accuracy and robustness. In this article, we introduce our approach that is based on a refinement of the set of initial partitions and uses differential evolution algorithm in order to find the most valid solution. Properties of the algorithm are demonstrated on four benchmark datasets.

Keywords: consensus clustering, differential evolution, ensemble, data

Classification: 62H30, 92G30

1. INTRODUCTION

In the age of growing data, it is necessary to think about techniques which help to reduce data in order to draw any useful conclusions from it. However, as data stored in databases are getting bigger and bigger, new techniques must arise simultaneously to face the computational problems.

Cluster analysis (or clustering) is a machine learning technique that deals with the data reduction problem. Based on similarity, it tries to split data into sets of classes (called clusters) such that objects in the same class are as similar as possible and objects in different classes are as dissimilar as possible. This process of data reduction is not new and is known from antiquity, when western philosophers Plato and Aristotle introduced the concept of grouping objects into categories based on their similar properties [15].

It is important to note that clustering is regarded as one of the most difficult tasks in machine learning, since neither number of classes nor true label of any object is often known a priori. Similar tasks are also called "unsupervised" comparing to "supervised" (e. g. classification), where true label of each object is known and the task is only to find the decision boundary between known classes.

Although first modern similarity and clustering ideas appeared in psychology and anthropology in the first half of the twentieth century [3], clustering as a new field was established mainly during 1950s and expanded especially during 1980s due to a massive spread of personal computers. Nowadays, clustering is an interdisciplinary area in science and new theoretical ideas can be found not only in mathematical or computer

DOI: 10.14736/kyb-2014-5-0661

science journals but also in biological. Moreover, practical applications of clustering are spread into miscellaneous branches of science, including natural language processing, market research, image processing or network analysis. With a bit of exaggeration, we can say that it is hard to find any area, where the idea of grouping similar object cannot be applied.

Since 1950s, many different clustering algorithms have been proposed. Some of the most important approaches are hierarchical [43], partitional [30], graph-based [53], mixture-model based [6], fuzzy [4], density-based [9], spectral [40] or subspace [1]. We also refer interest reader to any of these great review articles about clustering: [23, 24, 52].

We should also mention methods that use stochastic optimization strategies, although most of them are based on algorithms mentioned above. The first algorithm that combined clustering with an evolutionary algorithm (EA) was proposed by [38]. Since then, it has been shown by many authors that partitional methods give better performance when combined with EA [34]. In [29], authors propose a method called DE-KM, that combines k-means algorithm with differential evolution (DE). K-means algorithm is used to obtain the centroids for each initial solution in the DE population and also to fine-tune each new solution obtained by the mutation and crossover operators of the DE. In [36], authors compare the performance of partitional clustering combined with three EAs – DE, particle swarm optimization and genetic algorithm. They conclude that DE gives better results than other EAs. DE algorithm was also employed by [48]. Authors combine k-means with DE with competing strategies and conclude that hybrid variants using k-means algorithm for local search are much more efficient than non-hybrid DE algorithms. In [5], authors propose automatic clustering of large datasets with no prior knowledge of the true number of clusters. Again, their method is based on a combination of partitional clustering with EA. See also [22] for a comprehensive review of algorithms combining clustering and EAs.

Similarly as in supervised techniques, clustering also incorporated ensemble (or consensus) ideas, mainly in the last decade. Most of them can be roughly classified into three large groups – graph partitioning algorithms, algorithms based on coassociation matrix and probabilistic algorithms [13, 14, 45, 46].

From the first group, HGPA (Hypergraph Partitioning Algorithm) and MCLA (Meta-clustering Algorithm) were one of the first [45]. In HGPA, initial partitions are first transformed into a hypergraph that is subsequently partitioned by cutting a minimal number of hyperedges. Authors use the HMETIS package that searches for a minimal cut [26] . On the other hand, MCLA works with the so called meta-graph concept and is based on clustering clusters from the set of initial partitions. It is important to mention that HGPA and MCLA work only with clusters of similar sizes (otherwise they can result in trivial solution). Another graph partitioning approaches can be found in [10, 26].

A different idea is used in algorithms based on coassociation matrix. One of the first proposals, CSPA (Cluster based Similarity Partitioning Algorithm), also comes from [45]. For each initial clustering, a binary similarity matrix is created as follows. If two objects are in the same cluster, then corresponding entry is one, otherwise it is zero. This matrix is obtained for all partitions and then all matrices are averaged. The final matrix (whose entries can be interpreted as the fractions of clusterings in which two objects

are in the same cluster) is then used as an input into another clustering algorithm to produce final partition.

In probabilistic ensembles the labels are modeled as random variables drawn from a probability distribution defined as a mixture of multinomial component densities. Consensus clustering is then formulated as a maximum likelihood estimation problem [13, 46]. From other probabilistic approaches, we mention mainly Bayesian ensembles that can be found in [50] and [49].

There are also other approaches that can hardly be classified into one of the above groups, for example the approach in [8] which uses bootstrapped datasets to improve the accuracy of consensus partition.

The paper is organized as follows. First, in section 2, we define necessary terms about partitions. Differential evolution is described in section 3. Then, in section 4, we introduce our idea how to combine the set of clusterings into consensus solution. Experiments with benchmark datasets are performed in section 5 and section 6 concludes this article.

2. CLUSTERING

2.1. Lattice of partitions

This section is written mainly according to [41], where authors use an algebraic view on clustering. In addition, readers might be interested in various other mathematical tools for data mining methods described in this source. Unless stated otherwise, all sets in the following text are assumed to be finite.

Definition 2.1. (Partition of a set) Partition of a nonempty set M is set

$$\mathbf{C} = \{C_1, \ldots, C_k\} = \{C_i |\ i \in I\}$$

of k nonempty subsets of M such that $\bigcup_{i=1}^{k} C_i = M$ and $C_i \cap C_j = \emptyset$ for $i \neq j$. I is used here to denote an index set.

We will use Δ for the set of all partitions of set M. If partitions of more than one set will be discussed, we will use notation Δ_M to denote the set of all partitions of set M.

Definition 2.2. (Refinement) Let $\mathbf{C}_1 = \{C_i^{(1)} |\ i \in I\}, \mathbf{C}_2 = \{C_j^{(2)} |\ j \in J\} \in \Delta$ be two partitions. We will say that $\mathbf{C}_1$ is finer than $\mathbf{C}_2$ (what is denoted by $\mathbf{C}_1 \leq \mathbf{C}_2$) if every element in $\mathbf{C}_2$ is a union of elements of $\mathbf{C}_1$.

Theorem 2.3. (Trotter [47]) Ordered pair $(\Delta, \leq)$ (with refinement relation) is a partially ordered set.

Theorem 2.4. (Existence of infimum, Trotter [47]) If $\mathbf{C}_1 = \{C_i^{(1)} |\ i \in I\}, \mathbf{C}_2 = \{C_j^{(2)} |\ j \in J\} \in \Delta$ are two partitions, then the partition $\inf\{\mathbf{C}_1, \mathbf{C}_2\}$ (denoted by $\mathbf{C}_1 \wedge \mathbf{C}_2$) exists in the poset $(\Delta, \leq)$ and can be found as

$$\inf\{\mathbf{C}_1, \mathbf{C}_2\} = \{C_i^{(1)} \cap C_j^{(2)} |\ i \in I, j \in J \text{ and } C_i^{(1)} \cap C_j^{(2)} \neq \emptyset\}.$$

Corollary 2.5. (Lattice of partitions, Trotter [47]) Poset $(\Delta, \leq)$ from Theorem 2.3 is also a lattice.

Definition 2.6. (Stirling number of the second kind) Stirling number of the second kind, denoted as $S(n,k)$, is the total number of partitions of set with n objects into k subsets.

Definition 2.7. (Bell number) Bell number, denoted as $B(n)$, is the total number of partitions of set with n objects.

It can be shown (see [17]) that

$$S(n,k) = \frac{1}{k!}\sum_{j=0}^{k}(-1)^{k-j}\left(\begin{array}{c} k \\ j \end{array}\right) j^n$$

and therefore

$$B(n) = \sum_{k=0}^{n} S(n,k) = \sum_{k=0}^{n}\frac{1}{k!}\sum_{j=0}^{k}(-1)^{k-j}\left(\begin{array}{c} k \\ j \end{array}\right) j^n. \tag{1}$$

As noted by [17], for a fixed k, the number of partitions of a set with n elements into k subsets increases asymptotically exponentially. See Figure 2.1 for an example of all possible partitions of set with $n = 4$ elements.

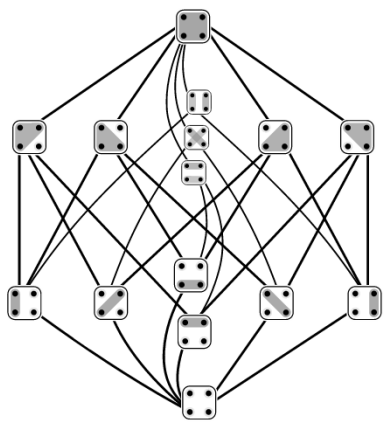

Fig. 1. Lattice of 15 possible partitions of set with 4 elements. Adapted from Wikipedia [51].

2.2. Consensus clustering

Many different clustering algorithms have been proposed. Natural question that immediately arises is if we can use information that comes from more than one partition of

dataset in order to produce a new and better partition. These approaches are called cluster ensembles (since ensemble of initial partitions is used) or consensus clustering.

Three main advantages, when compared with traditional clustering algorithms, are:

- an improved quality of partition (ability to produce a more valid partition),
- robustness (ability to deal with noisy data with outliers),
- parallelization (ability to integrate data from many sources).

Definition 2.8. (Clusterer, Strehl and Ghosh [45]) Any algorithm Φ that partitions M is called clusterer, i. e. $\Phi : M \to \mathbf{C} \in \Delta$.

Definition 2.9. (Consensus function, Strehl and Ghosh [45]) Let $\Lambda = \{\mathbf{C}_i\}_{i=1}^{r} \subset \Delta$ be set of r partitions of M obtained by r clusterers, where $\mathbf{C}_i = \{C_1^{(i)}, C_2^{(i)}, \ldots, C_{k_i}^{(i)}\}$ and let $\mathbf{C}_0 \in \Delta$. Then function $\Gamma : \Lambda \to \mathbf{C}_0$ is called a consensus function and $\mathbf{C}_0$ is called a consensus partition.

2.3. Partition validity

If data to be clustered is labeled, then the quality of any partition can be assessed simply by comparing true labels with labels assigned by the algorithm (these validity measures are called external). However, true labels are not known in many practical situations. Here, the only solution how to evaluate the quality of any partition is to infer it somehow from data itself. Measures used for these situations are therefore called internal. Most frequently used is the SSE criterion (sum of squared errors). This very simple index measures only the variability within clusters and takes the form

$$SSE(\mathbf{C}) = \sum_{C_k \in \mathbf{C}} \sum_{\mathbf{x}_i \in C_k} (\mathbf{x}_i - \boldsymbol{\mu}_k)^2 \in [0, \infty[,$$

where $\mathbf{x}_i$ is a feature vector of the $i-$th observation and $\boldsymbol{\mu}_k$ is the centre of cluster C_k, i. e.

$$\boldsymbol{\mu}_k = \frac{1}{|C_k|} \sum_{i:\mathbf{x}_i \in C_k} \mathbf{x}_i$$

SSE criterion is also the objective function for k-means – the most popular clustering algorithm (and one of the fastest). Obviously, for k fixed, lower values of criterion indicate a better partition. We do not use any other validity measure in this article, but we refer the interested reader to any of these reviews of validity measures used in literature [16, 19].

2.4. Clustering as optimization process

Definition 2.10. (Clustering as one dimensional optimization problem, Handl and Knowles [18]) Let $M = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ be set (called dataset) of n vectors (called objects). Let Δ be a set of all possible partitions of this set and let $V : \Delta \to \mathbb{R}$ be real

function (internal validity criterion). Clustering may then be defined as an optimization problem, i. e. to find

$$\operatorname*{argmin}_{\mathbf{C}\in\Delta} V(\mathbf{C}).$$

Remark 2.11. We can assume only the minimization task without a loss of generality (some internal validity indices need to be minimized while some maximized), since

$$\operatorname*{argmin}_{\mathbf{C}\in\Delta} V(\mathbf{C}) = \operatorname*{argmax}_{\mathbf{C}\in\Delta}\{-V(\mathbf{C})\}.$$

3. DIFFERENTIAL EVOLUTION

Optimization is any technique that tries to find the best element from some set, where "the best" is w.r.t. some criteria (i. e. functions). If there is only one criterion to be optimized, optimization is also called single-objective, whilst when optimizing based on more than one criterion, optimization is called multi-objective. Optimization methods are usually divided into two groups – deterministic (most common examples include Newton's method and Quasi-Newton's method) and stochastic (genetic algorithm, tabu search, hill climbing, etc.). The main advantage of using stochastic methods is that they often do not require optimized function to be differentiable, which allows them to solve a very wide range of problems.

The differential evolution (DE) algorithm (proposed by [44]) is a form of a stochastic optimization. When compared to the genetic algorithm, the DE is a much younger technique. In the following, the DE algorithm will be described w.r.t. a clustering problem to be solved.

Let $f : A \to \mathbb{R}$ be a function (called **cost**) that has to be minimized (or equivalently one can assume **fitness** function that has to be maximized). A is called **search space** (the set of all candidate solutions). In our problem, $A = \{1, 2, \ldots, n\}^n$ (see also Example 3.1). In other words, f takes label integer vector (whose length is n, i. e. the number of objects to be clustered) as input and returns real value (cluster validity criterion in our task). The task is to find

$$\arg\min_{\mathbf{w}\in A} f(\mathbf{w}),$$

i. e. such a partition of objects that is as valid as possible. In general, there may exist several such partitions.

We will denote ith **individual** in jth **generation** with $\mathbf{w}^{ij} = (w_1^{i,j}, w_2^{i,j}, \ldots, w_n^{i,j}) \in A$. Each generation consists of population of $Npop \geq 4$ individuals and let $G \geq 1$ denote the number of generations. The sequence of generations is also called **evolution** (that is why the method is called evolution). Let $F \in [0, 2]$ be a **mutation constant** and $CR \in [0, 1]$ a **crossover rate**. In summary, parameters $Npop, F, CR$ and G have to be set in advance.

Example 3.1. For $n = 3$ (three objects to be clustered) and $Npop = 6$, we have $A = \{(111), (112), (113), (121), (122), (123), (131), (132), (133), (211), (212), (213), (221), (222), (223), (231), (232), (233), (311), (312), (313), (321), (322), (323), (331), (332), (333)\}$ with $|A| = 3^3 = 27$ elements (label vectors). For example, the first element indicates

such a partition of the three-element set, where all objects are in one cluster only. Note that some elements in A represent the same partition. For example elements

$$(121), (212), (131), (313), (323), (232)$$

represent a partition, where the first and third object are in one cluster, while the second object comes from a different cluster. According to (1), there are exactly five different partitions of a set with three elements.

Since $Npop = 6$, $\{(121), (222), (333), (121), (312), (123)\}$ is an example of a population of six individuals that form some generation.

Given above parameters, the algorithm can be described as follows:

1. Set $Npop \geq 4, F \in [0, 2], CR \in [0, 1], G \geq 1$.

2. Generate randomly initial (null) generation $\{\mathbf{w}^{1,0}, \mathbf{w}^{2,0}, \ldots, \mathbf{w}^{Npop,0}\}$, i. e., in our task, $w_l^{i,0}$ are random numbers drawn randomly from $\{1, 2, \ldots, s\}$ $\forall\, l \in \{1, 2, \ldots, s\}$ and $\forall\, i \in \{1, 2, \ldots, Npop\}$.

3. For each generation $j \in \{1, 2, \ldots, G\}$ do

   (a) $\forall\, i \in \{1, 2, \ldots, Npop\}$ pick randomly 3 individuals $\mathbf{w}^{1,j}, \mathbf{w}^{2,j}, \mathbf{w}^{3,j}$ (distinct from $\mathbf{w}^{i,j}$) from population and create a new individual $\mathbf{z}^{i,j+1} = \lceil \mathbf{w}^{3,j} + F(\mathbf{w}^{2,j} - \mathbf{w}^{1,j}) \rceil$ ($\lceil\, \rceil$ denotes vectorized ceiling function[1]).

   (b) $\forall\, i \in \{1, 2, \ldots, Npop\}$ and $\forall\, l \in \{1, 2, \ldots, s\}$ set

$$v_l^{i,j+1} = \begin{cases} z_l^{i,j+1} & \text{if } (randb(l) \leq CR) \text{ or } l = rnbr(i) \\ w_l^{i,j} & \text{if } (randb(l) > CR) \text{ and } l \neq rnbr(i), \end{cases}$$

   where $randb(l)$ is the lth evaluation of a uniform random number generator with outcome from $[0, 1]$ and $rnbr(i)$ is a randomly chosen number from $\{1, 2, \ldots, s\}$. This ensures that $\mathbf{v}^{i,j+1}$ will have at least one parameter from $\mathbf{z}^{i,j+1}$ [44].

   (c) $\forall\, i \in \{1, 2, \ldots, Npop\}$ compare cost value of $\mathbf{v}^{i,j+1}$ with $\mathbf{w}^{i,j}$ and set

$$\mathbf{w}^{i,j+1} = \begin{cases} \mathbf{v}^{i,j+1} & \text{if } f(\mathbf{v}^{i,j+1}) < f(\mathbf{w}^{i,j}) \\ \mathbf{w}^{i,j} & \text{otherwise.} \end{cases}$$

4. From the final generation, pick the individual with the lowest cost value, i. e. solution of optimization task is

$$\mathbf{w}^* = \underset{u \in \{\mathbf{w}^{1,G}, \mathbf{w}^{2,G}, \ldots, \mathbf{w}^{Npop,G}\}}{\arg\min} f(u).$$

[1]For any real number x, the ceiling function is defined as $\lceil x \rceil = min\{a \in \mathbb{Z}, a \geq x\}$.

Remarks

- As can be seen in 3.a) and 3.b), if one sets $F = 0$ and $CR = 1$, then all generations will consist of the same population (no change in time). Moreover, values of F close to zero indicate that the vector $\mathbf{z}^{i,j+1}$ in 3.a) will be close to $\mathbf{w}^{3,j}$ and vice versa. The higher the value of F, the more different $\mathbf{z}^{i,j+1}$ will be when compared to $\mathbf{w}^{3,j}$. That is why this constant is called the mutation parameter. On the other hand, the parameter CR approximates the probability with which we replace the value $w_l^{i,j}$ with the value $z_l^{i,j+1}$. That is why it is called the crossover probability (although it is not the exact probability, since at least one mutation always occurs [33]).

- An optimal choice of parameters $F, CR, Npop$ and G is crucial, therefore it is very important to choose it carefully. Many rules of thumb exist, see for example [37].

- Thanks to step 3.c), it is guaranteed that each following generation consists of at least as good individuals as the previous one.

- It can easily happen (steps 2. and 3.a)) that new individuals do not lie in A. There are many solutions how to deal with this problem. We often assign boundary values or we can simply generate new ones.

4. CONSENSUS CLUSTERING WITH DIFFERENTIAL EVOLUTION

Here we propose our approach to the consensus clustering task based on three concepts: refinement of partitions, differential evolution and classification (see also table below for pseudocode). The algorithm has three parameters - δ (for adjusting the cardinality of the refinement), $\mathbf{h}$ (vector of possible values for number of clusters), and ϵ (for outlier detection). The number of clusters as well as outliers will be determined automatically.

Input: M (dataset), $\Lambda = \{\mathbf{C}_1, \mathbf{C}_2, \ldots, \mathbf{C}_r\}$ (a set of initial partitions), V (validity criterion), δ, $\mathbf{h}$, ϵ (parameters for algorithm), $Npop$, F, CR, G (parameters for differential evolution)

Step 1: $\mathbf{C}^{'} \leftarrow \wedge_{i=1}^{r} \mathbf{C}_i$

Step 2: $\mathbf{C}^{''} \leftarrow \{C_i^{'} \in \mathbf{C}^{'}; |C_i^{'}| \geq \delta\}$

Step 3: Find $\mathbf{C}_* = \{C_1^{(*)}, C_2^{(*)}, \ldots, C_k^{(*)}\}$ with k in $\mathbf{h}$ as a result of optimization of $\mathbf{C}^{''}$ with DE algorithm with parameters $Npop$, F, CR, G.

Step 4: If $\mathbf{C}^{'} \setminus \mathbf{C}^{''} \neq \emptyset$, classify each $C \in \mathbf{C}^{'} \setminus \mathbf{C}^{''}$ such that

$$\min_{i \in \{1,2,\ldots,k\}} dist(C, C_i^{(*)}) \leq \epsilon$$

to class $C_{\blacktriangledown}^{(*)}$, where $\blacktriangledown = \operatorname{argmin}_{i \in \{1,2,\ldots,k\}} dist(C, C_i^{(*)})$.

Output: Set of k clusters and set of outliers.

Tab. 1. Pseudocode of our algorithm.

Creating initial partition(s)

Note that the task in consensus clustering is not to propose a new approach on how to cluster objects, but instead to focus on how to combine existing partitions in order to get a better result (in other words, we search for a consensus function Γ). For this purpose, let

$$\{\mathbf{C}_1, \mathbf{C}_2, \ldots, \mathbf{C}_r\} = \Lambda \subset \Delta_M$$

be the set of initial partition(s), where

$$\mathbf{C}_i = \{C_1^{(i)}, C_2^{(i)}, \ldots, C_{k_i}^{(i)}\} \; \forall \, i \in \{1, 2, \ldots, r\}.$$

Note that we allow that it may consist of just one element.

Step 1 – Refinement of initial partitions

Due to Theorem 2.4, infimum exists in each lattice of partitions. Let us denote the infimum of partitions (this step is skipped in case of just one initial partition) from Λ (i. e. $\wedge_{i=1}^{r} \mathbf{C}_i$) by $\mathbf{C}^{'} = \{C_1^{'}, C_2^{'}, \ldots, C_t^{'}\} \in \Delta_M$. Obviously, $t \geq \max\{k_1, k_2, \ldots, k_r\}$. Therefore, one initial partition may have a large impact on "granularity" of $\mathbf{C}^{'}$ (and consequently also on validity). On the other hand, $\mathbf{C}^{'}$ has a highly desired property – it may reveal "homogenous" subsets from M (note that two objects are in the same cluster in $\mathbf{C}^{'}$ if they are in the same clusters in all initial partitions).

Step 2 – Thresholding

When clustering large datasets (situations when thousands of objects are partitioned) or when using a high number of initial partitions, $\mathbf{C}^{'}$ may result in a very high number of elements from which many can have a very small cardinality. Since our next step will be optimization, we have to decrease t as much as possible (for computational purposes). That is why we decided in this step to remove all elements from $\mathbf{C}^{'}$, whose cardinality is lower than a given threshold δ. More precisely, let

$$\{C_i^{'} \in \mathbf{C}^{'}; |C_i^{'}| \geq \delta\} = \{C_1^{''}, C_2^{''}, \ldots, C_s^{''}\} = \mathbf{C}^{''},$$

where

$$M \supseteq M^{'} = \{\mathbf{x} \in M; \exists j \in \{1, 2, \ldots, t\} : \mathbf{x} \in C_j^{'} \wedge |C_j^{'}| \geq \delta\}.$$

Obviously, $\mathbf{C}^{''}$ is a subset of $\mathbf{C}^{'}$, therefore $s \leq t$. When $\delta = 1$ (no elements are removed from $\mathbf{C}^{'}$), we have $\mathbf{C}^{''} = \mathbf{C}^{'}$ and $M^{'} = M$.

Step 3 – Optimization

Now we have only those (homogenous) elements from $\mathbf{C}^{'}$, whose cardinality is not less than the threshold. As mentioned, infimum of partitions may have a very bad validity because of its granularity. On the other hand, if we merged some elements from $\mathbf{C}^{''}$ now, we could get even more valid results than each of the initial partitions. That is

why in this step, our focus is on merging some clusters from $\mathbf{C}^{''}$ in order to increase its validity.

Note that we could also follow this approach with the original set of objects M or with $\mathbf{C}^{'}$. But in both cases, cardinalities of corresponding sets may still be too high and the optimization process would be too slow. Therefore, we decided to reduce the number of objects as much as possible and afterwards apply optimization in order to find the solution in a reasonable amount of time. On the other hand, there are also situations when only a small number of objects is to be partitioned. In this case, we can set $\delta = 1$ and no elements from $\mathbf{C}^{'}$ will be removed.

Now we use DE to merge some elements from $\mathbf{C}^{''}$. As in Example 3.1, we will use integer representation. For example, if we have $s = 6$ elements $\{C_1^{''}, C_2^{''}, \ldots, C_6^{''}\}$, then one possible partition of them is $\{C_1^{''} \cup C_5^{''}, C_2^{''} \cup C_3^{''} \cup C_4^{''}, C_6^{''}\}$ and it can be represented, for example, by an integer vector $(1, 2, 2, 2, 1, 3)$. As noted before, some optimization techniques are not suitable, since they have an assumption of continuous and differentiable objective function. We therefore suggest to use the DE algorithm, since it has no restriction on objective function. Let $\mathbf{C}_* \in \Delta_{M'}$ denote the partition returned by the optimization process and let k be the number of elements in this partition, i. e.

$$\mathbf{C}_* = \{C_1^{(*)}, C_2^{(*)}, \ldots, C_k^{(*)}\}.$$

If we have any prior information about any possible values for k, we may include it in vector $\mathbf{h}$, the second parameter of the algorithm. Optimization will then search only for those candidate solutions whose k is in $\mathbf{h}$.

Step 4 – Classification and outlier detection

Note that $\mathbf{C}_*$ is not always a partition of M. On the other hand, k (estimated number of clusters in M) is already known. In the last step, we have to focus on a classification of elements from $\mathbf{C}^{'} \setminus \mathbf{C}^{''}$ (if this is empty, procedure ends) to elements from $\mathbf{C}_*$.

It is important to note that we transformed unsupervised problem (unknown labels for all objects in the beginning of procedure) to supervised (classes are known and the task is to assign remaining elements to one of them). There are many proposals for solving supervised problems (AdaBoost, stacking, Naive-Bayes, decision trees). Maybe the simplest one can be the nearest neighbor algorithm, that classifies each $C \in \mathbf{C}^{'} \setminus \mathbf{C}^{''}$ to class $C_{\blacktriangledown}^{(*)}$, where

$$\blacktriangledown = \underset{i \in \{1,2,\ldots,k\}}{\operatorname{argmin}} \; dist(C, C_i^{(*)}).$$

Any distance measure can be used here. For example, we can use the distance measure from single-linkage hierarchical algorithm defined as

$$dist(C, C_i^{(*)}) = \min_{\mathbf{x} \in C, \mathbf{y} \in C_i^{(*)}} d(\mathbf{x}, \mathbf{y}),$$

where d is any distance function between two vectors. Note that we could also set another parameter here (let us denote it as ϵ) and classify only those objects whose distance to any class is less then ϵ. Otherwise, the object would be labeled as an outlier. See [9] for one possible idea how to find ϵ.

5. EXPERIMENTS

In this section we compare our algorithm to several other approaches proposed in literature. We use four benchmark datasets – *iris*, *wine*, *glass* and *vowel*. First three are from [2] and the last is from [35]. Together with many other datasets, these are widely used in literature when comparing performance of clustering algorithms. Basic characteristics of the data are below.

Dataset	# of objects	# of features	# of classes
iris	150	4	3
wine	178	13	3
glass	214	9	6
vowel	871	3	6

Tab. 2. Datasets used in experiments.

The objective is to minimize SSE criterion for the given number of clusters (ranging from two to six). 12 clustering algorithms are used to compare their performance to 6 variants of our proposal (denoted as CCDE). Brief information about each of them is given below. We divide all 18 algorithms into two groups – partitional and ensemble.

- **Partitional methods:**
  - k-means (labeled as P k-means [20])
  - partitioning around medoids (P PAM [27]) from package *cluster* by [31]
  - kernel k-means (P kernel k-means) and spectral clustering (P spectral) from *kernlab* package [25]
  - k-means with hard (P hardcl) and soft (P neuralgas) competitive learning implemented in *cclust* package [7]
  - spherical k-means with genetic algorithm (P spherical k-means [28]) implemented in package *skmeans* [21]
  - model-based clustering (P model-based [11]) implemented in *mclust* package [12]
- **Ensemble methods:**
  - four variants of consensus clustering with bootstrapped datasets (E k-means B, E PAM B, E k-means and E PAM [32]) implemented in package *clusterCons* [42]
  - six variants of our proposal (E CCDE 2-6 1, E CCDE 2-6 2, E CCDE 2-6 3, E CCDE k 1, E CCDE k 2 and E CCDE k 3)

In E k-means B and E k-means, 30 runs of k-means were used to create an ensemble. In E PAM B and E PAM, 30 runs of PAM algorithm were used instead. In E k-means B and E PAM B (E k-means and E PAM), 80% (100%) of observations was sampled (default value was 80%). We should stress here that algorithms E k-means B, E PAM B, E k-means and E PAM do not return a partition but only a (consensus) dissimilarity

	$\sqrt{SSE}$ k = 2	$\sqrt{SSE}$ k = 3	$\sqrt{SSE}$ k = 4	$\sqrt{SSE}$ k = 5	$\sqrt{SSE}$ k = 6	RT k = 2	RT k = 3	RT k = 4	RT k = 5	RT k = 6
P k-means	12.34	9.46	7.84	7.31	6.68	0.00	0.00	0.00	0.00	0.00
P PAM	12.38	8.88	7.61	6.87	6.53	0.00	0.00	0.00	0.00	0.01
P kernel k-means	14.34	11.41	10.15	9.98	10.11	0.08	0.09	0.10	0.11	0.14
P spectral	12.45	11.76	10.26	8.76	8.56	0.24	0.24	0.24	0.25	0.24
P hardcl	12.34	9.60	8.02	7.34	6.85	0.03	0.03	0.03	0.03	0.03
P neuralgas	12.34	8.88	7.57	7.06	6.30	0.04	0.05	0.06	0.07	0.08
P spherical k-means	12.45	9.68	9.44	9.31	9.26	2.11	4.79	7.62	11.47	11.96
P model-based	12.45	9.50	8.79	7.45	7.17	0.02	0.04	0.06	0.06	0.09
E k-means B	12.34	8.88	7.57	7.06	6.26	6.32	6.38	6.36	6.37	6.28
E PAM B	12.38	8.88	7.64	6.87	6.36	6.27	6.30	6.36	6.29	6.44
E k-means	12.34	8.88	7.76	7.09	6.57	6.06	6.31	6.22	6.36	6.49
E PAM	12.38	8.88	7.61	6.87	6.53	6.32	6.32	6.35	6.65	6.35
E CCDE 2-6 1	12.34	8.89	7.63	7.01	6.60	1.19	1.17	1.22	1.18	1.15
E CCDE 2-6 2	12.39	9.17	8.13	7.50	7.09	1.08	1.15	1.18	1.13	1.09
E CCDE 2-6 3	12.42	9.23	8.59	7.67	7.54	1.07	1.15	1.16	1.10	0.89
E CCDE k 1	12.34	8.88	7.57	8.06	9.79	0.77	0.96	1.11	1.42	1.58
E CCDE k 2	12.34	8.88	7.57	8.93	9.94	0.77	0.97	1.09	1.36	1.49
E CCDE k 3	12.34	8.88	8.15	8.77	10.53	0.74	0.97	1.06	1.42	1.40

Tab. 3. Squared roots of SSE and runtime in seconds (RT) in experiments with *iris* dataset. Minimal values of validity criterion are in grey.

	k = 2	k = 3	k = 4	k = 5	k = 6
P k-means	2131.61	1556.41	1154.55	990.84	824.04
P PAM	2141.92	1546.47	1154.08	1000.76	843.13
P kernel k-means	3724.83	3269.86	3239.77	2795.71	2368.24
P spectral	2543.80	2040.23	1744.56	1325.34	1182.17
P hardcl	2132.28	1552.00	1258.18	1036.33	856.63
P neuralgas	2131.67	1539.70	1154.89	964.81	825.97
P spherical k-means	3068.06	2115.42	1896.21	1603.86	1665.34
P model-based	2324.49	2247.16	2180.31	2143.72	1948.38
E k-means B	2131.61	1539.70	1154.62	1034.47	827.07
E PAM B	2141.92	1567.02	1154.08	984.30	843.13
E k-means	2132.08	1539.70	1158.18	1030.33	827.07
E PAM	2141.92	1546.47	1154.08	1000.76	843.13
E CCDE 2-6 1	2131.61	1540.06	1158.47	1004.39	862.22
E CCDE 2-6 2	2159.11	1555.43	1177.63	1015.18	871.64
E CCDE 2-6 3	2154.16	1635.80	1241.90	1109.46	960.53
E CCDE k 1	2131.61	1539.70	1154.08	1111.83	988.77
E CCDE k 2	2131.61	1539.70	1154.08	1253.14	1323.64
E CCDE k 3	2131.61	1762.24	1154.08	1350.70	1617.52

Tab. 4. Squared roots of SSE in experiments with *wine* dataset. Minimal values in each column are in grey.

	k = 2	k = 3	k = 4	k = 5	k = 6
P k-means	42.85	38.02	33.87	30.06	27.80
P PAM	43.05	38.35	36.19	30.26	27.47
P kernel k-means	53.86	51.25	49.65	48.69	47.72
P spectral	55.92	49.20	41.39	36.17	36.65
P hardcl	42.92	38.34	34.24	31.57	28.43
P neuralgas	42.85	38.04	31.93	29.92	27.40
P spherical k-means	42.86	37.56	31.95	29.44	27.18
P model-based	53.49	52.88	52.10	46.08	45.91
E k-means B	42.85	38.04	37.29	30.35	42.51
E PAM B	43.05	38.35	36.53	30.15	27.47
E k-means	42.85	38.05	37.29	30.17	42.51
E PAM	43.05	38.35	36.19	30.26	27.47
E CCDE 2-6 1	42.85	37.63	32.13	29.79	27.73
E CCDE 2-6 2	46.88	41.01	36.51	34.29	31.22
E CCDE 2-6 3	47.40	42.67	37.93	34.95	32.22
E CCDE k 1	42.85	37.54	32.61	30.91	30.59
E CCDE k 2	42.85	39.22	38.16	37.35	36.22
E CCDE k 3	42.85	40.25	37.83	40.97	38.85

Tab. 5. Squared roots of SSE in experiments with *glass* dataset. Minimal values in each column are in grey.

	k = 2	k = 3	k = 4	k = 5	k = 6
P k-means	9284.28	7935.20	6696.14	6058.82	5640.81
P PAM	9286.28	7939.67	6670.54	5983.78	5554.92
P kernel k-means	11526.43	8426.99	8024.35	7358.12	6939.43
P spectral	11576.69	8827.63	7240.97	6615.66	6116.20
P hardcl	9284.28	7934.17	6914.31	6142.33	5732.94
P neuralgas	9284.51	7927.30	6646.48	6086.22	5541.16
P spherical k-means	10178.46	8567.30	7704.28	7382.10	7147.77
P model-based	9601.63	9103.60	9335.81	9034.07	8497.00
E k-means B	9284.76	7936.76	6646.99	5979.80	5668.87
E PAM B	9286.28	7944.39	6661.18	5983.78	5564.71
E k-means	9284.28	7938.20	6646.96	5979.82	5739.45
E PAM	9286.28	7939.67	6672.17	5985.33	5555.15
E CCDE 2-6 1	9284.28	8022.26	6862.79	6243.66	5834.65
E CCDE 2-6 2	9409.87	8031.77	6932.89	6256.74	5957.58
E CCDE 2-6 3	9374.74	8013.43	6925.06	6351.36	5979.34
E CCDE k 1	9284.28	7923.13	6718.65	6769.22	9731.81
E CCDE k 2	9284.28	7922.58	6872.60	6948.38	10803.82
E CCDE k 3	9284.28	7944.51	6976.66	7012.49	11653.48

Tab. 6. Squared roots in SSE of experiments with *vowel* dataset. Minimal values in each column are in grey.

matrix. Ward's (minimum variance) method is then applied to find the final partition since it tries to minimize SSE criterion. Six variants of our algorithm can be divided into two groups – in variants E CCDE 2-6 1, E CCDE 2-6 2 and E CCDE 2-6 3, a set of initial partitions (Λ) consists of $r = 5$ k-means partitions for $k = 2, 3, 4, 5, 6$ (the only difference is the value of δ parameter that is 1, 2 and 3 respectively). On the other hand, in variants E CCDE k 1, E CCDE k 2 and E CCDE k 3, Λ is created from $r = 30$ runs of k-means with the same number of clusters (that is equal to the number of clusters to be found in corresponding experiment). Again, three values of δ parameter are used.

For all stochastic algorithms, 30 runs were performed and mean values of SSE were computed. 100 iterations were set in all iterative algorithms in order to compare runtime of each algorithm. For methods using the genetic algorithm or DE, population size was 10. Default values of other population parameters were used in DE ($F = 0.8, CR = 0.5$). In all other algorithms, default values of parameters were used also.

All experiments were performed with the *R* language by [39] on Intel(R) Core(TM) i7-3630QM CPU (2.4GHz) with 8 GB RAM. *DEoptim* package was used for DE algorithm [33]. Runtime was estimated only for one run of each algorithm. It is also very important to stress here that runtimes are only approximate since they depend also on implementations in corresponding R packages.

All results are reported in Tables 3, 4, 5 and 6. Instead of SSE, we report square roots. Runtime is estimated only for *iris* dataset.

As can be seen from tables, results are approximately consistent across all datasets.

As expected, ensemble methods overcome single clustering algorithms. On the other hand, although they give more valid partitions, they consume much more time.

In summary, E CCDE k 1, E CCDE k 2 and E k-means B achieved the best results. Final value of their objective was minimal (when compared to other approaches) in 9, 8 and 8 experiments respectively (of $4 * 5 = 20$ experiments in total). On the other hand, runtime of all E CCDE variants is lower than in other ensemble techniques. In general, three variants of E CCDE k gave better results than variants of E CCDE 2-6. Therefore, the careful choice of Λ has a large impact on the final partition. Another important corollary is the fact that our approach gives better results in situations with small k. The higher the k, the worse the results it gives. Moreover, when k is higher, E CCDE k variants give worse results than E CCDE 2-6 variants. It is also interesting to note that variants with $\delta > 1$ do not give better results than variants with $\delta = 1$ but as can be seen from Table 3, runtime of the algorithm decreases with an increasing δ.

6. CONCLUSION

The main goal for our research was to propose a new technique for how different partitions could be effectively merged. Similar algorithms are called consensus or ensemble in literature. Although these techniques are not new clustering algorithms, they provide frameworks on how to integrate information that comes from many sources into one final solution.

Our results show that our proposal, although consuming more time than single clustering methods, gives more valid solutions. The results are better when the number of clusters to be found is low. The parameter δ can be used to speed up the algorithm, since this can significantly decrease the number of elements in refinement. However, so far there is no rule how to determine it in advance.

It is very important to mention here that we have used only the classical strategy for DE (called *DE/rand/1/bin* in literature) with default parameters. There exist many improvements of DE and also advanced rules how to determine parameters of the algorithm to speed up its convergence. Similarly, set Λ was chosen in all experiments just by running k-means method many times. It is therefore a topic for another research, how to optimally choose the set of initial partitions. Results from supervised ensemble techniques suggest that the most accurate algorithms are based on the idea of independent learners, i. e. that members of ensembles should be as diverse as possible. However, this could result in too many elements in refinement, what is not desired.

Another important note is that in all our experiments, we do not use our algorithm for estimating the true number of clusters in data. The reason is that the minimum of any validity criterion does not automatically correspond to partition based on true labels. Another problems arise in noisy datasets with overlapping clusters that are very common in practice. However, in case of gaussian clusters that are well separated and the amount of noise is not high, it is also important to determine the number of components in mixture, i. e. the number of clusters. Therefore, another topics for future research are to find out how to deal with a higher number of clusters in data and how good is our algorithm at estimating the true number of clusters.

ACKNOWLEDGEMENT

This work was supported by the Slovak Research and Development Agency under the contracts No. APVV-0303-11 and No. APVV-0661-10 and also by VEGA 1/0143/11. The author would also like to thank the anonymous reviewers for their valuable comments and suggestions to improve the quality of the paper.

(Received July 31, 2013)

REFERENCES

[1] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan: Automatic subspace clustering of high dimensional data for data mining applications. In: Proc. 2001 ACM SIGMOD International Conference on Management of data *27* (1998), 2, pp. 94–105.

[2] K. Bache and M. Lichman: UCI machine learning repository, 2013. URL `http://archive.ics.uci.edu/ml`.

[3] K. D. Bailey: Typologies and Taxonomies: An Introduction to Classification Techniques. Sage Publications Inc., Los Angeles 1994.

[4] J. C. Bezdek: Pattern Recognition with Fuzzy Objective Function Algorithms. Plenum Press, New York 1981.

[5] S. Das, A. Abraham, and A. Konar: Automatic clustering using an improved differential evolution algorithm. IEEE Trans. Sys. Man Cyber., Part A: Systems and Humans *38* (2008), 1, 218–237.

[6] A. P. Dempster, N. M. Laird, and D. B. Rubin: Maximum likelihood from incomplete data via the em algorithm. J. Roy. Stat. Soc. Ser. B *39* (1977), 1, 1–38.

[7] E. Dimitriadou: cclust: Convex Clustering Methods and Clustering Indexes, 2012. URL `http://CRAN.R-project.org/package=cclust`.

[8] S. Dudoit and J. Fridlyand: Bagging to improve the accuracy of a clustering procedure. Bioinformatics *19* (2003), 9, 1090–2003.

[9] M. Ester, H. P. Kriegel, J. Sander, and X. Xu: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proc. 2nd International Conference on Knowledge Discovery and Data Mining 1996, pp. 226–231.

[10] X. Fern and C. Brodley: Solving cluster ensemble problems by bipartite graph partitioning. In: Proc. 21st International Conference on Machine learning 2004, pp. 36–43.

[11] C. Fraley and A. E. Raftery: Model-based clustering, discriminant analysis and density estimation. J. Amer. Statist. Assoc. *97* (2002), 611–631.

[12] C. Fraley and A. E. Raftery: MCLUST Version 3 for R: Normal Mixture Modeling and Model-Based Clustering. Techn. Report 504, University of Washington, Department of Statistics, 2006.

[13] R. Ghaemi, N. Sulaiman, H. Ibrahim, and N. Mustapha: A survey: Clustering ensembles techniques. In: Proc. International Conference on Computer, Electrical, and Systems Science, and Engineering (CESSE) *38* (2009), pp. 644–653.

[14] J. Ghosh and A. Acharya: Cluster ensembles. Wiley Interdisc. Rew.: Data Mining and Knowledge Discovery *1* (2011), 4, 305–315.

[15] S. J. Gould: Full House: The Spread of Excellence from Plato to Darwin. Harmony, New York 1996.

[16] M. Halkidi, Y. Batistakis, and M. Vazirgiannis: Cluster validity methods: Part i. SIGMOD Record *31* (2002), 2, 40–45.

[17] J. Handl and J. Knowles: Multi-objective clustering and cluster validation. In: Multi-Objective Machine Learning (Studies in Computational Intelligence, Vol, 16), Springer, Berlin 2006, pp. 21–47.

[18] J. Handl and J. Knowles: An evolutionary approach to multiobjective clustering. IEEE Trans. Evolutionary Comput. *11* (2007), 56–76.

[19] J. Handl, J. Knowles, and D. Kell: Computational cluster validation in post-genomic data analysis. Bioinformatics *21* (2005), 15, 3201–3212.

[20] J. Hartigan and M. Wong: A k-means clustering algorithm. Applied Statistics *28* (1979), 100–108.

[21] K. Hornik, I. Feinerer, M. Kober, and C. Buchta: Spherical *k*-means clustering. J. Statist. Software *50* (2012), 10, 1–22.

[22] E. Hruschka, R. Campello, A. Freitas, and A. de Carvalho: A survey of evolutionary algorithms for clustering. IEEE Trans. Sys. Man Cyber. Part C: Applications and Reviews *39* (2009), 2, 133–155.

[23] A. K. Jain: Data clustering: 50 years beyond k-means. Pattern Recognition Lett. *31* (2010), 8, 651–666.

[24] A. K. Jain, M. N. Murty, and P. J. Flynn: Data clustering: A review. ACM Comput. Surveys *31* (1999), 3, 316–323.

[25] A. Karatzoglou, A. Smola, K. Hornik, and A. Zeileis: kernlab – an S4 package for kernel methods in R. J. Statist. Software *11* (2004), 9, 1–20.

[26] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar: Multilevel hypergraph partitioning: Applications in vlsi domain. In: Proc. Design and Automation Conference, 1997, pp. 526–529.

[27] L. Kaufman and P. Rousseeuw: Finding Groups in Data: An Introduction to Cluster Analysis. Wiley, New York 1990.

[28] K. Krishna and M. Narasimha Murty: Genetic k-means algorithm. Trans. Sys. Man Cyber. Part B *29* (1999), 3, 433–439.

[29] W. Kwedlo: A clustering method combining differential evolution with the k-means algorithm. Pattern Recognition Letters *32* (2011), 12, 1613–1621.

[30] J. MacQueen: Some methods for classification and analysis of multivariate observations. In: Proc. Fifth Berkeley Symposium on Mathematical Statistics and Probability *1* (1967), pp. 281–297.

[31] M. Maechler, P. Rousseeuw, A. Struyf, M. Hubert, and K. Hornik: cluster: Cluster Analysis Basics and Extensions, 2013. R package version 1.14.4.

[32] S. Monti, P. Tamayo, J. Mesirov, and T. Golub: Consensus clustering: A resampling-based method for class discovery and visualization of gene expression microarray data. Mach. Learn. *52* (2003), 1-2, 91–118.

[33] K. Mullen, D. Ardia, D. Gil, D. Windover, and J. Cline: DEoptim: An R package for global optimization by differential evolution. J. Statist. Software *40* (2011), 6, 1–26.

[34] C. Murthy and N. Chowdhury: In search of optimal clusters using genetic algorithms. Pattern Recognition Lett. *17* (1996), 8, 825–832.

[35] S. K. Pal and D. D. Majumder: Fuzzy sets and decision making approaches in vowel and speaker recognition. IEEE Trans. Sys. Man Cyber. *7* (1977), 625–629.

[36] S. Paterlini and T. Krink: Differential evolution and particle swarm optimisation in partitional clustering. Comput. Statist. Data Anal. *50* (2006), 5, 1220–1247.

[37] K. V. Price, R. M. Storn, and J. A. Lampinen: Differential Evolution: A Practical Approach to Global Optimization. Springer-Verlag, Berlin 2006.

[38] V. Raghavan and K. Birchand: A clustering strategy based on a formalism of the reproductive process in a natural system. In: Proc. Second International Conference on Information Storage and Retrieval, 1979, pp. 10–22.

[39] R Core Team. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna 2012. URL `http://www.R-project.org/`.

[40] J. Shi and J. Malik: Normalized cuts and image segmentation. In: IEEE Trans. Pattern Analysis and Machine Intelligence *22* (2000), 8, 888–905.

[41] D. A. Simovici and Ch. Djeraba: Mathematical Tools for Data Mining: Set Theory, Partial Orders, Combinatorics. Advanced information and knowledge processing. Springer, London 2008.

[42] T. I. Simpson, J. D. Armstrong, and A. P. Jarman: Merged consensus clustering to assess and improve class discovery with microarray data. BMC Bioinform. *11* (2010), 11–590.

[43] P. H. Sneath: The application of computers to taxonomy. Journal of general microbiology *17* (1957), 1, 201–226.

[44] R. Storn and K. Price: Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. J. Global Optim. *11* (1997), 4, 341–359.

[45] A. Strehl and J. Ghosh: Cluster ensembles – a knowledge reuse framework for combining partitionings. In: Proc. 11th National Conference On Artificial Intelligence, NCAI, Edmonton, Alberta 2002, pp. 93–98.

[46] A. Topchy, A. Jain, and W. Punch: A mixture model of clustering ensembles. In: Proc. SIAM International Conference on Data Mining 2004, pp. 22–24.

[47] W. M. Trotter: Combinatorics and Partially Ordered Sets. The Johns Hopkins University Press, Baltimore 1992.

[48] J. Tvrdík and I. Křivý: Differential evolution with competing strategies applied to partitional clustering. Lecture Notes Comput. Sci. *7269* (2012), 136–144.

[49] P. Wang, C. Domeniconi, and K. Laskey: Nonparametric bayesian clustering ensembles. Lecture Notes Comput. Sci. *6323* (2010), 3, 435–450.

[50] H. Wang, H. Shan, and A. Banerjee: Bayesian cluster ensembles. Stat. Anal. Data Min. *4* (2011), 1, 54–70.

[51] Wikipedia: Partition of a set. `http://en.wikipedia.org/wiki/Partition_of_a_set`.

[52] R. Xu and D. Wunsch: Survey of clustering algorithms. IEEE Trans. Neural Networks *16* (2005), 3, 645–678.

[53] Ch. T. Zahn: Graph-theoretic methods for detecting and describing gestalt clusters. IEEE Trans. Comput. *20* (1971), 31, 68–86.

Miroslav Sabo, Department of Mathematics and Descriptive Geometry, Faculty of Civil Engineering, Slovak University of Technology, Radlinského 11, 813 68 Bratislava. Slovak Republic.

e-mail: sabo@math.sk