Centroid Update Approach to K-Means Clustering

Ioan-Daniel BORLEA, Radu-Emil PRECUP, Florin DRAGAN, Alexandra-Bianca BORLEA
Departament of Automation and Applied Informatics, Politehnica University of Timisoara,
Bd. V. Parvan 2, 300223 Timisoara, Romania
idborlea@isoftwaresystems.com, radu.precup@upt.ro, florin.dragan@upt.ro,
abarbu@isoftwaresystems.com

***Abstract*—The volume and complexity of the data that is generated every day increased in the last years in an exponential manner. For processing the generated data in a quicker way the hardware capabilities evolved and new versions of algorithms were created recently, but the existing algorithms were improved and even optimized as well. This paper presents an improved clustering approach, based on the classical k-means algorithm, and referred to as the centroid update approach. The new centroid update approach formulated as an algorithm and included in the k-means algorithm reduces the number of iterations that are needed to perform a clustering process, leading to an alleviation of the time needed for processing a dataset.**

***Index Terms*—clustering algorithms, clustering methods, data analysis, data mining, machine learning algorithms.**

I. Introduction

Data mining is a domain that has developed in the last years as one of the most advanced ones in artificial intelligence [1]. Data mining processes large volumes of data for extracting valuable information, for finding patterns or for creating statistics, and uses different types of algorithms for processing datasets. Every dataset can be processed in several ways using different data mining processing algorithms depending on the information that needs to be extracted from it.

Clustering algorithms [2] belong to the data mining ones, and divide a dataset in smaller chunks of data called clusters. A cluster contains dataset records that have similarities and is obtained by running an iterative algorithm over a dataset for a specified number of times. Giving the fact that a clustering algorithm is an iterative process, the volume of time that is needed for splitting a processed dataset in clusters is increasing with the size of the processed dataset. For increasing the speed of a clustering process, there are two approaches: run a parallel clustering on multiple machines or using an optimized clustering algorithm. The resulting clusters can be post processed to obtain further information or statistics.

This paper suggests a partition-based clustering algorithm based on the k-means clustering algorithm [3], [4]. The proposed algorithm is an improved version of the k-means algorithm that reduces the number of iterations (cycles) needed for obtaining the final clusters. By reducing the number of iterations that are needed for running the clustering algorithm, the total processing time needed for obtaining the final clusters will also be reduced, meaning that the volume of data that can be processed by the same computing device in the same amount of time can be increased.

The rest of this paper is organized as follows: Section II provides an overview of the classical k-means clustering algorithm. Section III discusses some of the related work that has been performed in this domain. Section IV describes the new version of k-means algorithm that was proposed. Section V presents the obtained results using the new version of k-means algorithm. The conclusions, advantages, limitation and a future research are presented in Section VI.

II. Overview of the Classical K-Means Algorithm

The k-means algorithm [3], [4] is an unsupervised learning algorithm that divides a dataset in clusters and it is one of the most used algorithms for data mining [5]. The k-means algorithm is able to process only numerical values, so if a dataset that contains non-numerical value needs to be processed, a convert operation needs to be performed to bring all dataset vectors (records) $\mathbf{x}_i$ to the form of a *d*–dimensional set:

$$\{\mathbf{x}_i = [x_{i1} \quad x_{i2} \quad ... \quad x_{id}]^T \in \Re^d \mid i = 1...n\}, \tag{1}$$

where the superscript *T* stands for matrix transposition and *n* is the number of database records.

The objective of k-means algorithm is represented by minimizing the intra-cluster variance by solving the optimization problem:

$$\mathbf{c}^* = \arg\min_{\mathbf{c}\in\Re^{dk}} V(\mathbf{c}),\ V(\mathbf{c}) = \sum_{j=1}^{k} \sum_{\substack{i=1 \\ \mathbf{x}_i \in C_j}}^{n} \| \mathbf{x}_i - \mathbf{c}_j \|, \tag{2}$$

where: k – the number of clusters C_j, $j = 1...k$,

$\mathbf{c}_j = [c_{j1} \quad c_{j2} \quad ... \quad c_{jd}]^T \in \Re^d$ – the centroid of cluster C_j,

$\mathbf{c} = [\mathbf{c}_1^T \quad \mathbf{c}_2^T \quad ... \quad \mathbf{c}_k^T]^T \in \Re^{dk}$ – the centroid vector,

$\{\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_k\} \subset \Re^d$ – the set of centroids,

$\mathbf{c}^* = [(\mathbf{c}_1^*)^T \quad (\mathbf{c}_2^*)^T \quad ... \quad (\mathbf{c}_p^*)^T]^T \in \Re^{dk}$ – the vector of optimal centroids and also the solution to the optimization problem (2).

The k-means algorithm uses a mathematical formula for calculating the distance $d_{\mathbf{x}_i,\mathbf{c}_j}$ from every record $\mathbf{x}_i$ and a centroid $\mathbf{c}_j$. If the Euclidian distance $d_{\mathbf{x}_i,\mathbf{c}_j}$ is considered:

$$d_{\mathbf{x}_i,\mathbf{c}_j} = \| \mathbf{x}_i - \mathbf{c}_j \| = \sqrt{\sum_{l=1}^{d} (x_{il} - c_{jl})^2}, \tag{3}$$

Digital Object Identifier 10.4316/AECE.2017.04001

by replacing the distance formula in (2) with the formula for calculating the Euclidian distance, the optimization problem becomes:

$$\mathbf{c}^* = \arg\min_{\mathbf{c}\in\Re^{dk}} V(\mathbf{c}),\ V(\mathbf{c}) = \sum_{j=1}^{k}\sum_{\substack{i=1\\ \mathbf{x}_i\in C_j}}^{n}\sqrt{\sum_{l=1}^{d}(x_{il}-c_{jl})^2}. \quad (4)$$

The formula included in (4) that is used for calculating the distance between every record $\mathbf{x}_i$ and all centroids $\mathbf{c}_j$ determines the shape of the final clusters.

The k-means algorithm consists of five major steps, 0 to 4, presented as follows. The step 0 is executed once, and the other ones are executed in an iterative (cyclic) manner until the final clusters are generated.

Step 0. This is an initialization step, where the number of clusters *p*, the initial centroids, the maximum number of iterations and the accepted error are chosen. The general notation μ will be used as follows for the current iteration, $\mu = 1...\mu_{\max}$, where $\mu_{\max}$ is the maximum number of iterations, which needs to be initialized. The initial value $\mu = 1$ will be used.

There are several ways to initialize the centroid vector $\mathbf{c}$. This paper will use an initial centroid vector $\mathbf{c}^{(1)}$ that contains the points where we believe that the final centroids will be. This initial centroid vector is:

$$\mathbf{c}^{(1)} = [\mathbf{c}_1^{T(1)} \quad \mathbf{c}_2^{T(1)} \quad ... \quad \mathbf{c}_p^{T(1)}]^T \in \Re^{dp}, \quad (5)$$

and it will initialize the set of initial centroids:

$$\{\mathbf{c}_1^{(1)}, \mathbf{c}_2^{(1)}, ..., \mathbf{c}_p^{(1)}\} \subset \Re^d. \quad (6)$$

The accepted error $\varepsilon > 0$ needs to be initialized and it will be used as a stop condition in the algorithm.

Step 1. The distance $d_{\mathbf{x}_i,\mathbf{c}_j}$ between every vector $\mathbf{x}_i$ and a centroid $\mathbf{c}_j$ of cluster C_j needs to be calculated using (3) for $i = 1...n$ and $j = 1...k$.

Step 2. Based on the distances calculated in step 1, a new set of clusters can be generated. Each new cluster $C_j^{(\mu)}$ will contain the vectors $\mathbf{x}_m$ that are closer to its center, the centroid $\mathbf{c}_j^{(\mu)}$. The cluster generation process is based on the distances that were computed in step 1. Each point $\mathbf{x}_m$ has a number of *k* distances $d_{\mathbf{x}_m,\mathbf{c}_i^{(\mu)}}, i = 1...k$ that were computed, the vector $\mathbf{x}_m$ will be included into a cluster $C_j^{(\mu)}$ if the distance $d_{\mathbf{x}_m,\mathbf{c}_j^{(\mu)}}$ is minimum, which means:

$$C_j^{(\mu)} = \{\mathbf{x}_m \mid d_{\mathbf{x}_m,\mathbf{c}_j^{(\mu)}} \le d_{\mathbf{x}_m,\mathbf{c}_i^{(\mu)}},\ \forall i = 1...k\}, \quad (7)$$

where every vector $\mathbf{x}_m$ is assigned only to a single cluster $C_j^{(\mu)}$.

Step 3. The new centroid $\mathbf{c}_j^{(\mu+1)}$ needs to be computed for each cluster $C_j^{(\mu)}$ based on the vectors $\mathbf{x}_i$ that are assigned to the cluster $C_j^{(\mu)}$. The new centroid is computed on the basis of the next formula that calculates the average of the coordinates for all vectors $\mathbf{x}_j \in C_j^{(\mu)}$:

$$\mathbf{c}_j^{(\mu+1)} = \frac{1}{|C_j^{(\mu)}|}\sum_{\mathbf{x}_i\in C_j^{(\mu)}}\mathbf{x}_i,\ j = 1...p, \quad (8)$$

where $|C_j^{(\mu)}|$ is the number of vectors assigned to cluster $C_j^{(\mu)}$. Equation (8) generates the next centroid vector:

$$\mathbf{c}^{(\mu+1)} = [\mathbf{c}_1^{T(\mu+1)} \quad \mathbf{c}_2^{T(\mu+1)} \quad ... \quad \mathbf{c}_p^{T(\mu+1)}]^T \in \Re^{dp}. \quad (9)$$

The scalar version of (8) is:

$$c_{jl}^{(\mu+1)} = \frac{1}{|C_j^{(\mu)}|}\sum_{\mathbf{x}_i\in C_j^{(\mu)}} x_{il},\ l = 1...d,\ j = 1...p. \quad (10)$$

Step 4. The stop condition of the algorithm is checked:

$$|\mathbf{c}^{(\mu+1)} - \mathbf{c}^{(\mu)}| < \varepsilon. \quad (11)$$

If the condition (11) is fulfilled, this means that the k-means algorithm had run in a cyclic manner until the centroid vector $\mathbf{c}$ did not change between two consecutive cycles and the solution to the optimization problem (2) is the last obtained vector $\mathbf{c}$:

$$\mathbf{c}^* = \mathbf{c}^{(\mu+1)}, \quad (12)$$

which corresponds to the final centroids:

$$\{\mathbf{c}_1^*, \mathbf{c}_2^*, ..., \mathbf{c}_k^*\} = \{\mathbf{c}_1^{(\mu+1)}, \mathbf{c}_2^{(\mu+1)}, ..., \mathbf{c}_k^{(\mu+1)}\} \subset \Re^d. \quad (13)$$

If the condition (11) is not fulfilled, then an increment of the current iteration replacing μ with $\mu+1$ and the algorithm will continue with step 1.

III. Related Work

In the last years the applications that were developed for processing large volumes of data have grown. There are various domains that use algorithms for extracting valuable information from datasets. There were developed lots of applications for analyzing databases in different domains such the healthcare system, where big data application were developed for predicting a disease or making statistics [6], [7], finance [8], astronomy, where a lot of information is generated and needs to be processed [9], [10], transportation [11] or image processing [12].

The k-means algorithm can be applied to small datasets, medium datasets and even on big datasets. The total processing time that k-means needs to find an acceptable solution is proportional with the size of the processed dataset. If the hardware components that are used for running the k-means algorithm are not considered, it can be concluded that the total processing time that the k-means algorithm needs to process a dataset depends on two major things: the position of the initial centroids and the number of the iterations that are performed until an acceptable solution is found.

Because of the importance of initial centroid selection, different methods were proposed in the literature for improving the k-means by picking wisely the initial centroids. The k-means++ algorithm was proposed in [13] in order to speed up the performance of the k-means algorithm. The k-means++ algorithm picks the first centroid in a random manner as k-means algorithm does, but the others centroids are picked by the probability proportional to the shortest distance from all existing centroids. A disadvantage of the k-means++ algorithm is the fact that it is inherently sequential. Another version of k-means algorithm was suggested in [14], where the initial set of clusters is generated by computing pair-wise distances and all the closer points create a cluster. There is a threshold value for the minimum number of the points that can form a cluster. The initial centroids will result as the mean value of these

centroids. Another method for selecting the initial centroids presented in [15] deals with sorting all processed records in a table, splitting the table in multiple sections (number of section equals the number of desired centroids), and the mean value of each section will represent an initial centroid.

Another way of improving the performance of k-means algorithm is to reduce the number of the computation steps that are performed until an acceptable solution is found. The solution given in [16] to speed up the k-means algorithm consists in storing information about the data that is clustered. By keeping information about closer centroid for every processed record, many of the distance calculations can be avoided. A disadvantage of this algorithm is the fact that it does not perform well when high dimension vectors are processed. Others versions of the k-means algorithm were modified to use the triangle inequality [17], [18] to avoid the unnecessary point-center distance calculation. This algorithm caches $O(nk)$ distance bounds for reducing the number of distance calculation. Every time when a centroid moves, the cached distances are updated based on the triangle inequality. Another way to increase the speed of the k-means algorithm was proposed in [19] by observing the fact that in some cases, some of the clusters were not changing anymore between two consecutive iterations. For all points that are assigned to these static clusters, the distance calculation can be skipped.

Another direction that evolved into the last years is represented by the online clustering that uses fuzzy logic [22]–[38] for classifying a dataset in a non-iterative manner. This kind of applications is suitable for online data that cannot be loaded or stored into the memory for applying iterative clustering algorithms.

IV. New K-Means Algorithm with Centroid Update Approach

The proposed version of k-means algorithm has as a main target maintaining the objective specified in (2) but reducing the number of cycles that are performed for splitting all records $\mathbf{x}_i$ in clusters. For achieving this objective a new step will be added between steps 3 and 4 of the classical algorithm and it has as a main objective the estimation of the way of how the centroids will evolve based on their previous values.

A. Monitoring the Centroids Evolution

The k-means algorithm stops when the centroids $\{\mathbf{c}_1^{(\mu)}, \mathbf{c}_2^{(\mu)}, ..., \mathbf{c}_k^{(\mu)}\}$ are not changing anymore between two consecutive iterations. If a centroid $\mathbf{c}_j$ is considered along with its value at the iterations μ and $\mu-1$, then based on $\mathbf{c}_j^{(\mu)}$ and $\mathbf{c}_j^{(\mu-1)}$ the increment of centroid $\mathbf{c}_j$ at the iteration μ is defined as the vector $\Delta\mathbf{c}_j^{(\mu)}$:

$$\Delta\mathbf{c}_j^{(\mu)} = \mathbf{c}_j^{(\mu)} - \mathbf{c}_j^{(\mu-1)} = [\Delta c_{j1}^{(\mu)} \quad \Delta c_{j2}^{(\mu)} \quad ... \quad \Delta c_{jd}^{(\mu)}]^T \in \Re^d, \tag{14}$$
$$\Delta c_{jl}^{(\mu)} = c_{jl}^{(\mu)} - c_{jl}^{(\mu-1)}, l = 1...d, j = 1...k.$$

Each element of the increment vector $\Delta\mathbf{c}_j^{(\mu)}$ provides information about the way of how the centroid $\mathbf{c}_j$ evolved between two consecutive iterations of the algorithm:

- If the l^{th} element of the increment vector $\Delta\mathbf{c}_j^{(\mu)}$ equals zero $\Delta c_{jl}^{(\mu)} = 0$ between two consecutive iterations of the algorithm the l^{th} element of centroid $\mathbf{c}_j^{(\mu)}$ did not change from the previous iteration $\mathbf{c}_j^{(\mu-1)}$, namely $c_{jl}^{(\mu)} = c_{jl}^{(\mu-1)}$.
- If the l^{th} element of the increment vector $\Delta\mathbf{c}_j^{(\mu)}$ does not equal zero $\Delta c_{jl}^{(\mu)} \neq 0$ between two consecutive iterations of the algorithm the l^{th} element of centroid $\mathbf{c}_j^{(\mu)}$ has changed and did not match the one from the previous iteration $\mathbf{c}_j^{(\mu-1)}$, i.e. $c_{jl}^{(\mu)} \neq c_{jl}^{(\mu-1)}$.

If the evolution of the increment of a centroid is analyzed, it is concluded that every element of the increment vector has a bigger value at the beginning. In the end, when an acceptable solution $\{\mathbf{c}_1^*, \mathbf{c}_2^*, ..., \mathbf{c}_k^*\}$ is found, the values of the elements of the increment vector are zero.

B. Using the Evolution of the Centroids to Force the Convergence of the K-Means Algorithm Based on Centroid Update Approach

The evolution of the centroids can be used for forcing the convergence of the k-means algorithm to a solution.

The most convenient case when a centroid update approach can be applied is the case when all elements of the increment vector $\Delta\mathbf{c}_j^{(\mu)}$ of a centroid taken in absolute values are decreasing from the beginning of the algorithm until the algorithm ends, and every element of the increment vector keeps the same sign between consecutives cycles of the algorithm until the algorithm completes.

The worst situation for applying a centroid update approach is the case when all elements of the increment vector of a centroid taken in absolute value are making an increasing and decreasing between consecutive steps of the algorithm from the beginning of the algorithm until the algorithm ends, and every element of the increment vector changes its sign between consecutives cycles of the algorithm.

The centroid update approach can be applied if some of the following conditions are met:

a. The last four centroids $\mathbf{c}_j^{(\mu)}, \mathbf{c}_j^{(\mu-1)}, \mathbf{c}_j^{(\mu-2)}$ and $\mathbf{c}_j^{(\mu-3)}$ are known for $j = 1...k$.
b. The centroid update approach is applied if every element that has the same position index in the vectors $\Delta\mathbf{c}_j^{(\mu)}$, $\Delta\mathbf{c}_j^{(\mu-1)}$ and $\Delta\mathbf{c}_j^{(\mu-2)}$ has the same sign.
c. The centroid update approach is applied as long as a stop condition is not met.
d. The centroid update approach can be applied to one or multiple centroids in the same cycle of the algorithm.
e. The formula used to update the value of a centroid $\mathbf{c}_j^{(\mu)}$ is:

$$\mathbf{c}_j^{(\mu)} = [c_{j1}^{(\mu)} \quad c_{j2}^{(\mu)} \quad ... \quad c_{jd}^{(\mu)}]^T + \mathbf{M}[\Delta c_{j1}^{(\mu)} \quad \Delta c_{j2}^{(\mu)} \quad ... \quad \Delta c_{jd}^{(\mu)}]^T, j = 1...k, \tag{15}$$

where $\mathbf{M} = \text{diag}(a_1, a_2, ..., a_d)$, is a diagonal update matrix and $a_1, a_2, ..., a_d$ are parameters that have positive values. The updated centroid will be used in the next iteration (cycle) of the k-means algorithm, speeding up the way of how the processed record are assigned to centroid $\mathbf{c}_j$.

f. After the centroid update approach is applied in one iteration, the next iteration of the k-means algorithm needs to run without applying the centroid update approach because the algorithm needs to evaluate the effect that the centroid update approach that was applied in the previous iteration has produced.

C. The Stop Condition in the Centroid Update Approach

The current version of the proposed algorithm uses one of the following conditions as a stop condition in the centroid update approach:

I. A constant $\beta > 0$ multiplied with the value of the accepted update error $\varepsilon > 0$. This condition is applied in the case when the absolute values of the elements for each of the centroids are relative small. Every element of the increment vector is checked and if one of the elements of the increment vector considered as absolute value is lower than the updated parameter $\beta > 0$ multiplied by $\varepsilon > 0$, then the centroid update approach is not performed.

II. A percentage report between every element of the increment vector and the current value of the element in the same position for a centroid that is used to calculate the relative increment $|\frac{\Delta c_{ji}^{(\mu)}}{c_{ji}^{(\mu)}}| \cdot 100 > \varepsilon$. This condition is used in the case when the numeric elements of every centroid taken as absolute values are big. If this absolute ratio is lower than a predefined percent, for which in this paper the predefined value equals ε, then the centroid update approach is not performed.

D. Centroid Update Approach as an Algorithm

The centroid update approach can be summarized as an algorithm, referred to as centroid update algorithm, with the following steps:

0. The value of the accepted update error $\varepsilon > 0$ and the value of update parameter $\beta > 0$ are initialized based on the experience of the specialist.
1. If $\mu > 3$ and in previous step no centroid update was applied

Then

Go to step 2

Else

Go to step 3

2. Identify all centroids $\mathbf{c}_j^{(\mu)}$, $j = 1...k$, that respect the following conditions:

a. $\text{sgn}(\Delta c_{ji}^{(\mu)}) = \text{sgn}(\Delta c_{ji}^{(\mu-1)}) = \text{sgn}(\Delta c_{ji}^{(\mu-2)}), i = 1...d,$

b. $|\Delta c_{ji}^{(\mu)}| \geq \beta\varepsilon$ (condition I) or

$|\frac{\Delta c_{ji}^{(\mu)}}{c_{ji}^{(\mu)}}| \cdot 100 > \varepsilon, i = 1...d$ (condition II).

If there are centroids $\mathbf{c}_j^{(\mu)}$ that respect the conditions a and b

Then

Apply update approach to $\mathbf{c}_j^{(\mu)}$ using (15)

Else

Go to step 3.

3. Centroid update algorithm not applicable at this iteration.

V. Validation

For validating the proposed algorithm, multiple synthetic datasets were used. The synthetic datasets were created using a random generation process, and every dataset that was used for validating the algorithm had different types of records (the number of the elements of each record was between 2 and 4) and different ranges for the values in the dataset (1) [20].

The platform used to process the datasets using the k-means with centroid update algorithm is BigTim [21]. This platform allows running the k-means algorithm in a single thread or in a multithread manner. The BigTim platform works on Windows operating systems. Currently the platform runs the k-means algorithm on a single machine in a single or multiple thread version, but in the future it will be improved to support parallel computation on different machines.

The machine that was used to run the k-means algorithm has the following hardware components: CPU Intel Core i7 4510U dual core, 16 GB of DDR3 RAM memory at 800MHz, 256 SSD hard drive. The hardware components are not very important in this study because the number of cycles that were executed is of interest in this paper and not the execution time of the k-means algorithm.

The operating system that was used for validating the algorithm using the BigTim platform was Windows 10 and the build version was 10.0.10586.

The parameters $\beta = 10$ and $\varepsilon = 0.0001$ were used for all datasets that were processed to validate the algorithm.

A. Synthetic Dataset with Records of Two Scalars

The first dataset consists of 10 million records, each record representing a vector with two numerical elements ($d = 2$), $\{\mathbf{x}_i = [x_{i1} \quad x_{i2}]^T \in \Re^2 \mid i = 1...10000000\}$. The values of x_{i1} belong to $[-500, 200]$ and the values of x_{i2} belong to $[-100, 500]$.

Four records were chosen as initial centroids, $\mathbf{c}_1^{(1)} = [-400 \quad -80]^T$, $\mathbf{c}_2^{(1)} = [-200 \quad 50]^T$, $\mathbf{c}_3^{(1)} = [0 \quad 300]^T$, $\mathbf{c}_4^{(1)} = [50 \quad 400]^T$, and the maximum number of iterations was set to $\mu_{\max} = 200$.

The diagonal update matrix $\mathbf{M} = \text{diag}(a_1, a_2)$ was set such that $a_1 = a_2 \geq 0$, and the parameter $a_1 = a_2$ is referred to as the correction factor. The classical k-means algorithm was applied first to this dataset, and this corresponds to the zero value of the diagonal update matrix, i.e. $\mathbf{M} = \text{diag}(0,0)$ and the correction factor also set to zero, $a_1 = a_2 = 0$. The k-

means algorithm with centroid update approach was applied to the same dataset for five times using for each element of the diagonal matrix the values that belong to the set $a_1 = a_2 \in \{0.5,1,1.5,2,2.5\}$. The number of elements of the final clusters that were obtained by using the classical k-means algorithm and the k-means algorithm with centroid update approach remained unmodified.

The dependency between the number of iterations and the correction factor is presented in Fig. 1. The number of iterations was reduced for the first dataset from 43 (in the initial case, when the classical k-means algorithm was used) to 27.

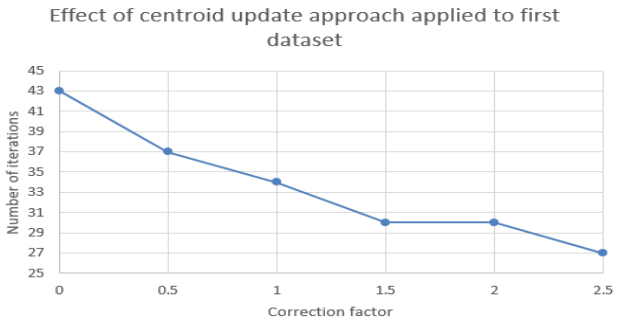

Figure 1. Number of iterations versus correction factor for the first dataset

The values that are picked for the correction factor have a direct impact over the total number of centroid updates that are performed by the algorithm, and this affects each of the centroids. If the correction factor that is used by the algorithm has a small value, then the convergence of the algorithm will not be so fast and the number of the centroid updates that are applied to each centroid will be larger. If the correction factor that is used by the algorithm has a big value, then the convergence of the algorithm will be quicker and the number of the centroid updates that are applied to each centroid will decrease.

The dependency between the number of centroid updates performed over the four centroids by the centroid update algorithm in some of the cycles of the k-means algorithm and the correction factor $a_1 = a_2 \in \{0,0.5,1,1.5,2,2.5\}$ (the correction factor equals zero, $a_1 = a_2 = 0$, in the case when the classical k-means algorithm is used) is presented in Fig. 2.

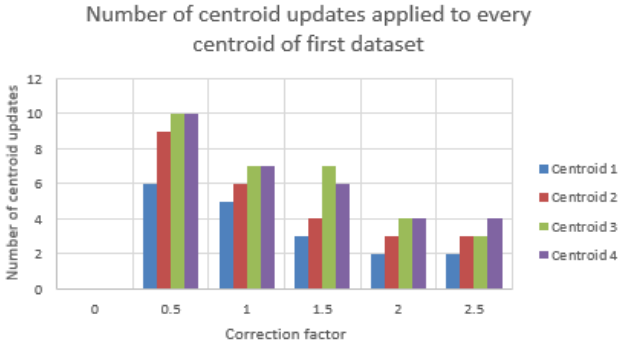

Figure 2. Number of centroid updates versus correction factor for the first dataset

The dependency between the improvement of the convergence speed measured as a percent and the correction factor is presented in Fig. 3. The maximum improvement has been obtained for the correction factor set to 2.5 which implies a reduction of 37.21% of the number of iterations performed.

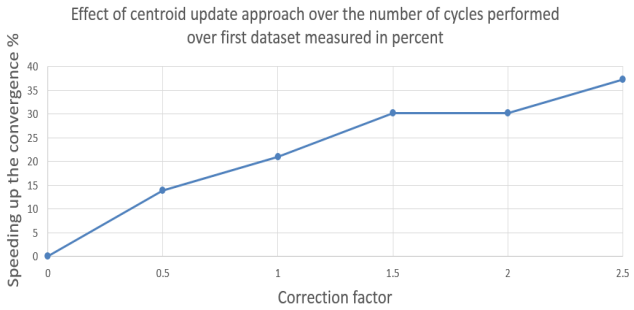

Figure 3. Reduction of the number of iterations measured as percentage for the first dataset

B. *Synthetic Dataset with Records of Three Scalars*

The second dataset consists of 10 million records, each record representing a vector with three numerical elements ($d = 3$), $\{\mathbf{x}_i = [x_{i1} \quad x_{i2} \quad x_{i3}]^T \in \Re^3 \mid i = 1...10000000\}$. The values of x_{i1} belong to $[-1000,200]$, the values of x_{i2} belong to $[-1000,1000]$ and the values of x_{i3} belong to $[-300,1000]$.

Four records were chosen as initial centroids, $\mathbf{c}_1^{(1)} = [-800 \quad -800 \quad -100]^T$, $\mathbf{c}_2^{(1)} = [-200 \quad 100 \quad -200]^T$, $\mathbf{c}_3^{(1)} = [300 \quad -400 \quad 500]^T$, $\mathbf{c}_4^{(1)} = [-300 \quad 800 \quad 700]^T$, and $\mu_{max} = 200$ was set.

The diagonal update matrix $\mathbf{M} = \text{diag}(a_1, a_2, a_3)$ was set such that $a_1 = a_2 = a_3 \geq 0$, and the parameter $a_1 = a_2 = a_3$ is again referred to as the correction factor as in the previous two applications. The classical k-means algorithm was applied first to this dataset, and this corresponds to the zero value of the diagonal update matrix, i.e. $\mathbf{M} = \text{diag}(0,0,0)$ and the correction factor also set to zero, $a_1 = a_2 = a_3 = 0$. The k-means algorithm with centroid update approach was applied to the same dataset for five times using for each element of the diagonal matrix the values $a_1 = a_2 = a_3 \in \{0.25,0.5,0.75,1,1.25\}$. The number of elements of the final clusters that were obtained by using the classical k-means algorithm, and the k-means algorithm with centroid update approach remained unmodified for this dataset as well.

The dependency between the number of iterations and the correction factor is presented in Fig. 4. The number of iterations was reduced for the second dataset from 38 (in the initial case, when the classical k-means algorithm was used) to 31.

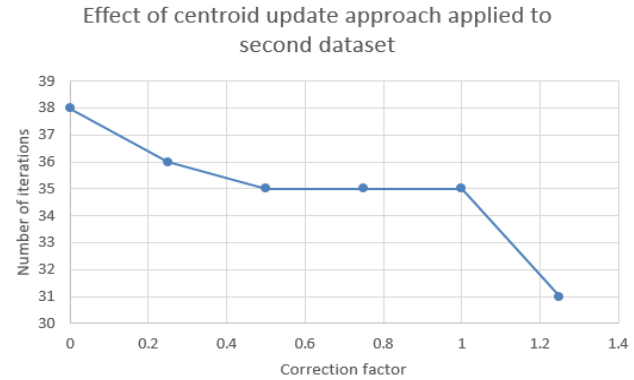

Figure 4. Number of iterations versus correction factor for the second dataset

The number of centroid updates that are performed over the centroids respect the same rule as for the first dataset: the number of the centroid updates decreases if the correction factor increases. The dependency between the number of centroid updates performed over the four centroids by the centroid update algorithm in some of the cycles of the algorithm and the correction factor $a_1 = a_2 = a_3 \in \{0,0.25,0.5,0.75,1,1.25\}$ (the correction factor equals zero, $a_1 = a_2 = a_3 = 0$, in the case when the classical k-means algorithm is used) is presented in Fig. 5.

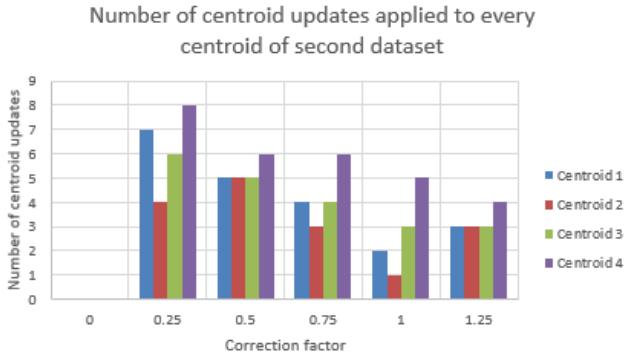

Figure 5. Number of centroid updates versus correction factor for the second dataset

The dependency between the improvement of the convergence speed measured as a percent and the correction factor is presented in Fig. 6. The maximum improvement has been obtained for the correction factor set to 1.25 which implies a reduction of 18.42% or the number of iterations performed.

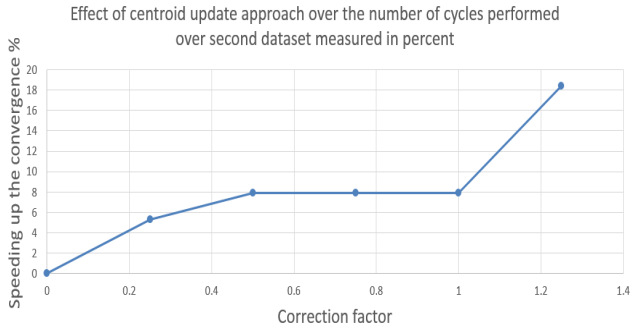

Figure 6. Reduction of the number of iterations measured as percentage for the second dataset

C. Synthetic Dataset with Records of Four Scalars

The third dataset consists of 10 million records, each record representing a vector with four numerical elements ($d = 4$), $\{\mathbf{x}_i = [x_{i1} \quad x_{i2} \quad x_{i3} \quad x_{i4}]^T \in \Re^4 \mid i = 1...10000000\}$. The values of x_{i1} belong to $[-1000,1000]$, the values of x_{i2} belong to $[-1000,1000]$, the values of x_{i3} belong to $[-1000,1000]$ and the values of x_{i4} belong to $[-1000,1000]$.

Four records were chosen as initial centroids:

$$\begin{aligned} \mathbf{c}_1^{(1)} &= [-900 \quad -500 \quad -100 \quad -100]^T, \\ \mathbf{c}_2^{(1)} &= [-700 \quad -300 \quad 0 \quad 200]^T, \\ \mathbf{c}_3^{(1)} &= [-450 \quad -160 \quad 70 \quad 400]^T, \\ \mathbf{c}_4^{(1)} &= [-300 \quad 0 \quad 30 \quad 800]^T, \end{aligned} \tag{16}$$

and $\mu_{max} = 200$ was set once more.

The diagonal update matrix $\mathbf{M} = \text{diag}(a_1, a_2, a_3, a_4)$ was set such that $a_1 = a_2 = a_3 = a_4 \geq 0$, and the parameter $a_1 = a_2 = a_3 = a_4$ is referred to as the correction factor. The classical k-means algorithm was applied first to this dataset, and this corresponds to the zero value of the diagonal update matrix, i.e. $\mathbf{M} = \text{diag}(0,0,0,0)$ and the correction factor also set to zero, $a_1 = a_2 = a_3 = a_4 = 0$. The k-means algorithm with centroid update approach was applied to the same dataset for four times using for each element of the diagonal matrix the values that belong to the set $a_1 = a_2 = a_3 = a_4 \in \{0.25,0.5,0.75,1\}$. For this dataset, the number of elements of the final clusters that were obtained by using the classical k-means algorithm, and the k-means algorithm with centroid update approach did not remain the same. The number of the elements from each of the clusters had changed for different values of the correction factor ($a_i, i = 1..4$) as shown in Table I. The quality of the resulting clusters is not affected, the numbers of elements of each of the final clusters obtained by using the centroid update algorithm are actually changed only by a maximum difference of three elements, and if the difference would be reported in percent, the number of elements of each cluster obtained by the k-means algorithm with centroid update approach differs with only $0.000001 - 0.000002\%$ with respect to the number of elements of the corresponding clusters obtained by the classical k-means algorithm.

TABLE I. THE NUMBER OF ELEMENTS IN EACH CLUSTER AS FUNCTION OF THE CORRECTION FACTOR

	Cluster 1	**Cluster 2**	**Cluster 3**	**Cluster 4**
$a_i = 0$	2376792	2119321	2557611	2946276
$a_i = 0.25$	2376792	2119321	2557611	2976276
$a_i = 0.5$	2376792	2119321	2557611	2946276
$a_i = 0.75$	2376790	2119324	2557613	2946273
$a_i = 1$	2376789	2119324	2557612	2946275

The dependency between the number of iterations and the correction factor is presented in Fig. 7. The number of iterations was reduced for the third dataset from 131 (in the initial case, when the classical k-means algorithm was used) to 103.

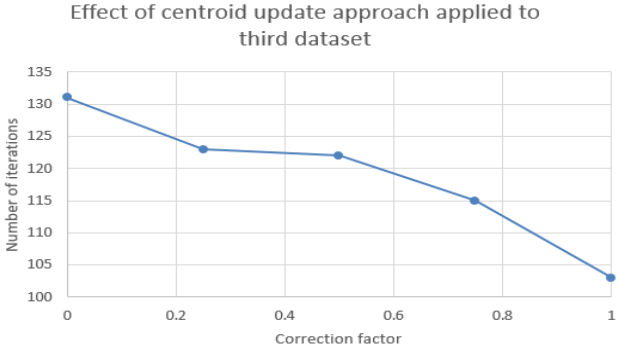

Figure 7. Number of iterations versus correction factor for the third dataset

The number of centroid updates that are performed over the centroids respects the same rule as for the first and second datasets, namely: the number of centroids updates decreases if the correction factor increases. The dependency between the number of centroid updates performed over the four centroids by the centroid update algorithm in some of the cycles of the algorithm and the correction factor

$a_1 = a_2 = a_3 = a_4 \in \{0.25, 0.5, 0.75, 1\}$ (the correction factor equals zero, $a_1 = a_2 = a_3 = a_4 = 0$, in the case when the classical k-means algorithm is used) is presented in Fig. 8.

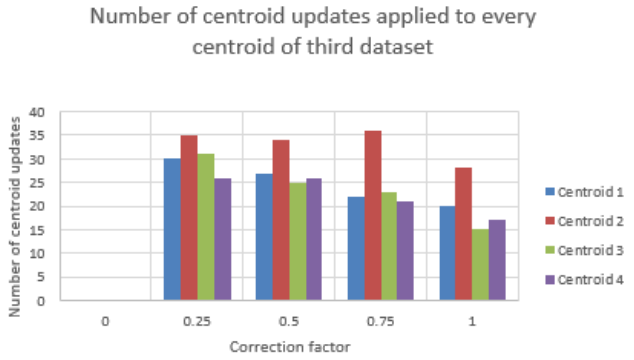

Figure 8. Number of centroid updates versus correction factor for the third dataset

The dependency between the improvement of the convergence speed measured as a percent and the correction factor is presented in Fig. 9. The maximum improvement has been obtained for the correction factor set to 1 which implies a reduction of 21.37% for the number of iterations performed.

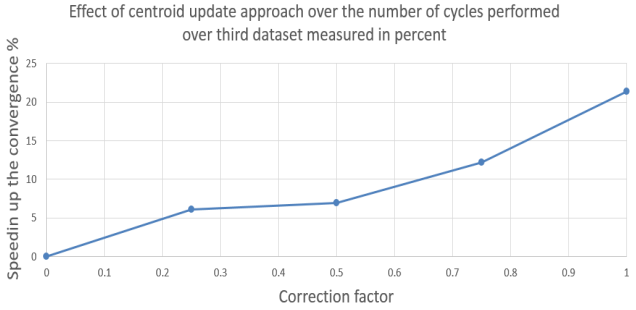

Figure 9. Reduction of the number of iterations measured as percentage for the third dataset

VI. Conclusion

This paper has proposed an approach to speed up the k-means algorithm by reducing the number of the iterations that are performed until an acceptable solution is found for the optimization criterion. This approach has been integrated in a new k-means algorithm that was developed using the classical version of the k-means algorithm, where a supplementary step was introduced for estimating the evolution of the centroids. The additional step speeds up the clustering process by updating the centroids in each step of the algorithm if some conditions are met.

Giving the fact that all algorithms perform well on a specific area, the centroid update approach inserted in the k-means algorithm has advantages and also disadvantages. The main benefit of the presented approach is the fact that it reduces the number of steps that are needed to obtain the final clusters. For the three datasets that were processed using this algorithm the number of cycles needed for obtaining the final clusters was reduced in average with 25.67% if the best scenario is considered.

The centroid update approach is recommended to be applied to large datasets, where it reduces the number of iterations in some cases to half (comparing it with the classical k-means algorithm). A disadvantage of this algorithm is represented by the fact that if the convergence of the k-means algorithm is forced too much using high values of the diagonal update matrix $\mathbf{M}$ elements, then the final clusters will be different from the ones that are obtained using the classical k-means algorithm. The centroid update algorithm tries to provide a faster way for obtaining the final clusters by reducing the computation time, but at the same time it tries to maintain the quality of the resulting clusters. For validating the results the content of all the resulting clusters obtained with the proposed algorithm was checked record by record, and contains the same records that were contained in the clusters obtained by using the classical k-means approach.

Future research will be dedicated to the update factor and the stop condition, which can be improved. Currently the partial update of a centroid (only some of its elements) is under research in order to understand the impact that will have over the other clusters, and this will make the approach to be also applicable to the databases that have multidimensional records with higher elements. In addition, the improvement of the optimization algorithms to solve the optimization problem (4) will be treated in terms of using several classical and modern optimization algorithms [22]–[38], but modified such that to work with integer variables. In some of the cases when the update factor and the stop conditions are not wisely chosen, the results of the classical k-means algorithm will differ from the results that are obtained using the centroid update approach presented in this paper.

Acknowledgment

This work was supported by grants of the Partnerships in priority areas – PN II program of the Executive Agency for Higher Education, Research, Development and Innovation Funding (UEFISCDI), project numbers PN-II-PT-PCCA-2013-4-0544 and PN-II-PT-PCCA-2013-4-0070, and UEFISCDI, project number PN-II-RU-TE-2014-4-0207.

References

[1] C. Eaton, P. Zikopoulos, T. Deutsch, G. Lapis, and D. Deroos, Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data. New York: McGraw-Hill, 2012.

[2] A. Fahad, N. Alshatri, Z. Tari, A. Alamri, I. Khalil, A. Y. Zomaya, S. Foufou, and A. Bouras, "A survey of clustering algorithms for big data: taxonomy and empirical analysis," IEEE Trans. Emerg. Top. Comput., vol. 2, no. 3, pp. 267–279, Sep. 2014. doi:10.1109/TETC.2014.2330519

[3] J. Mac Queen, "Some methods for classification and analysis of multivariate observations," in Proc. Fifth Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, CA, USA, 1967, vol. 1, pp. 281–297.

[4] S. Lloyd, "Least squares quantization in PCM," IEEE Trans. Inf. Theory, vol. 28, pp. 129–137, Mar. 1982. doi:10.1109/TIT.1982.1056489

[5] X. Wu, V. Kumar, J. R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. F. M. Ng, B. Liu, P. S. Yu, Z.-H. Zhou, M. Steinbach, D. J. Hand, and D. Steinberg, "Top 10 algorithms in data mining," Knowl. Informat. Syst., vol. 14, no. 1, pp. 1–37, Jan. 2008. doi:10.1007/s10115-007-0114-2

[6] J. Andreu-Perez, C. C. Y. Poon, R. D. Merrifield, S. T. C. Wong, and G.-Z. Yang, "Big data for health," IEEE J. Biomed. Health Inform., vol. 19, no. 4, pp. 1193–1208, July 2015. doi:10.1109/JBHI.2015.2450362

[7] S. Ram, W. L. Zhang, M. Williams, and Y. Pengetnze, "Predicting asthma-related emergency department visits using big data," IEEE J. Biomed. Health Inform., vol. 19, no. 4, pp. 1216–1223, July 2015. doi:10.1109/JBHI.2015.2404829

[8] W. Breymann, A. Dias, and P. Embrechts, "Dependence structures for multivariate high-frequency data in finance," Quant. Finance, vol. 3, no. 1, pp. 1–14, 2003. doi:10.1080/713666155

[9] P. Dewdney, P. Hall, R. Schilizzi, and J. Lazio, "The square kilometre array," Proc. IEEE, vol. 97, no. 8, pp. 1482–1496, Aug. 2009. doi:10.1109/JPROC.2009.2021005

[10] C. Reed, D. Thompson, W. Majid, and K. Wagstaff, "Real time machine learning to find fast transient radio anomalies: A semi-supervised approach combining detection and RFI excision," in Proc. International Astronomical Union Symposium on Time Domain Astronomy, 2011, pp. 1–6.

[11] J. Erman, M. Arlitt, and A. Mahanti, "Traffic classification using clustering algorithms," in Proc. SIGCOMM Workshop on Mining Network Data, Pisa, Italy, 2006, pp. 281–286.

[12] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, "An efficient k-means clustering algorithm: analysis and implementation," IEEE Trans. Pattern Anal. Mach. Intell., vol. 24, no. 7, pp. 881–892, Jul. 2002. doi:10.1109/TPAMI.2002.1017616

[13] D. Arthur and S. Vassilvitskii, "k-means++: the advantages of careful seeding," in Proc. Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, New Orleans, LA, USA, 2007, pp. 1027–1035.

[14] S. Nasser, R. Alkhaldi, and G. Vert, "A modified fuzzy k-means clustering using expectation maximization," in Proc. 2006 IEEE International Conference on Fuzzy Systems, Vancouver, BC, Canada, 2006, pp. 231–235. doi:10.1109/FUZZY.2006.1681719

[15] K. A. A. Nazeer, S. D. M. Kumar, and M. P. Sebastian, "Enhancing the k-means clustering algorithm by using a O(n logn) heuristic method for finding better initial centroids," in Proc. 2011 Second International Conference on Emerging Applications of Information Technology, Washington, DC, USA, 2011, pp. 261–264. doi:10.1109/EAIT.2011.57

[16] D. Pelleg and A. Moore, "Accelerating exact k-means algorithms with geometric reasoning," in Proc. ACM SIGKDD Fifth International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, 1999, pp. 277–281.

[17] C. Elkan, "Using the triangle inequality to accelerate k-means," in Proc. Twentieth International Conference on Machine Learning, Washington, DC, USA, 2003, pp. 147–153.

[18] A. W. Moore, "The anchors hierarchy: using the triangle inequality to survive high dimensional data," in Proc. Twelfth Conference on Uncertainty in Artificial Intelligence, CA, USA, 2000, pp. 397–405.

[19] T. Kaukoranta, P. Franti, and O. Nevalainen, "A fast exact GLA based on code vector activity detection," IEEE Trans. Image Process., vol. 9, no. 8, pp. 1337–1342, Aug. 2000. doi:10.1109/83.855429

[20] https://drive.google.com/open?id=0B4h3SEPCfIblM0NhTFJBMl9aazQ.

[21] I.-D. Borlea, R.-E. Precup, and F. Dragan, "On the architecture of a clustering platform for the analysis of big volumes of data," in Proc. IEEE 11th International Symposium on Applied Computational Intelligence and Informatics, Timisoara, Romania, 2016, pp. 145–150. doi:10.1109/SACI.2016.7507361

[22] P. Baranyi, D. Tikk, Y. Yam, and R. J. Patton, "From differential equations to PDC controller design via numerical transformation," Comput. Ind., vol. 51, no. 3, pp. 281–297, Aug. 2003. doi:10.1016/S0166-3615(03)00058-7

[23] I. Škrjanc, S. Blažič, and O. E. Agamennoni, "Identification of dynamical systems with a robust interval fuzzy model," Automatica, vol. 41, no. 2, pp. 327–332, Feb. 2005. doi:10.1016/j.automatica.2004.09.010

[24] F. G. Filip, "Decision support and control for large-scale complex systems," Annual Rev. Control, vol. 32, no. 1, pp. 61–70, Apr. 2008. doi:10.1016/j.arcontrol.2008.03.002

[25] D. Martín, R. Del Toro, R. Haber, and J. Dorronsoro, "Optimal tuning of a networked linear controller using a multi-objective genetic algorithm and its application to one complex electromechanical process," Int. J. Innov. Comput. Informat. Control, vol. 5, no. 10 (B), pp. 3405–3414, Oct. 2009.

[26] J. Vaščák and M. Paľa, "Adaptation of fuzzy cognitive maps for navigation purposes by migration algorithms," Int. J. Artif. Intell., vol. 8, no. S12, pp. 20–37, Oct. 2012.

[27] R.-E. Precup, R.-C. David, E. M. Petriu, S. Preitl, and M.-B. Radac, "Novel adaptive charged system search algorithm for optimal tuning of fuzzy controllers," Expert Syst. Appl., vol. 41, no. 4, pp. 1168–1175, Mar. 2014. doi:10.1016/j.eswa.2013.07.110

[28] D. Wijayasekara, O. Linda, M. Manic, and C. G. Rieger, "Mining building energy management system data using fuzzy anomaly detection and linguistic descriptions," IEEE Trans. Ind. Informat., vol. 10, no. 3, pp. 1829–1840, Aug. 2014. doi:10.1109/TII.2014.2328291

[29] R.-E. Precup, M.-C. Sabau, and E. M. Petriu, "Nature-inspired optimal tuning of input membership functions of Takagi-Sugeno-Kang fuzzy models for anti-lock braking systems," Appl. Soft Comput., vol. 27, pp. 575–589, Feb. 2015. doi:10.1016/j.asoc.2014.07.004

[30] A. Y. Jaen-Cuellar, L. Morales-Velazquez, R. Romero-Troncoso, and R. A. Osornio-Rios, "FPGA-based embedded system architecture for micro-genetic algorithms applied to parameters optimization in motion control," Adv. Electr. Comput. Eng., vol. 15, no. 1, pp. 23–32, Mar. 2015. doi:10.4316/AECE.2015.01004

[31] O. Arsene, I. Dumitrache, and I. Mihu, "Expert system for medicine diagnosis using software agents," Expert Syst. Appl., vol. 42, no. 4, pp. 1825–1834, Mar. 2015. doi:10.1016/j.eswa.2014.10.026

[32] A. Basgumus, M. Namdar, G. Yilmaz, and A. Altuncu, "Performance comparison of the differential evolution and particle swarm optimization algorithms in free-space optical communications systems," Adv. Electr. Comput. Eng., vol. 15, no. 3, pp. 17–22, Sep. 2015. doi:10.4316/AECE.2015.02003

[33] A. Moharam, M. A. El-Hosseini, and H. A. Ali, "Design of optimal PID controller using NSGA-II algorithm and level diagram," Stud. Informat. Control, vol. 24, no. 3, pp. 301–308, Sep. 2015.

[34] E. Osaba, E. Onieva, F. Dia, R. Carballedo, P. Lopez, and A. Perallos, "A migration strategy for distributed evolutionary algorithms based on stopping non-promising subpopulations: A case study on routing problems," Int. J. Artif. Intell., vol. 13, no. 2, pp. 46–56, Oct. 2015.

[35] J. K. Tar, J. F. Bitó, and I. J. Rudas, "Contradiction resolution in the adaptive control of underactuated mechanical systems evading the framework of optimal controllers," Acta Polyt. Hung., vol. 13, no. 1, pp. 97–121, Jan. 2016. doi:10.12700/APH.13.1.2016.1.8

[36] S. B. Ghosn, F. Drouby, and H. M. Harmanani, "A parallel genetic algorithm for the open-shop scheduling problem using deterministic and random moves," Int. J. Artif. Intell., vol. 14, no. 1, pp. 130–144, Mar. 2016.

[37] C. I. González, P. Melin, J. R. Castro, O. Castillo, and O. Mendoza, "Optimization of interval type-2 fuzzy systems for image edge detection," Appl. Soft Comput., vol. 47, pp. 631–643, Oct. 2016. doi:10.1016/j.asoc.2014.12.010

[38] A. Fakharian and R. Rahmani, "An optimal controlling approach for voltage regulation and frequency stabilization in islanded microgrid system," Control Eng. Appl. Informat., vol. 18, no. 4, pp.107–114, Dec. 2016.