Proceedings of the 2017 International Conference on Machine Learning and Cybernetics, Ningbo, China, 9-12 July, 2017

ADAPTIVE USAGE OF *K*-MEANS IN EVOLUTIONARY OPTIMIZED DATA CLUSTERING

XI WANG[1], WEIGUO SHENG[2,*]

[1]School of Computer Science and Technology, Zhejiang University of Technology, Hangzhou 310023, P.R.China
[2]Department of Computer Science, Hangzhou Normal University, Hangzhou 310036, P.R.China
E-MAIL: w.sheng@ieee.org

Abstract:

Evolutionary algorithm hybridizing with *k*-means operation has been widely employed for data clustering. The *k*-means operation in this approach, however, is generally applied with a fixed number of iteration and at each generation (i.e., fixed intensity and frequency) during evolution, which could be far more than optimal. In this paper, we first introduce a generalized *k*-means usage framework, which can be used to arbitrary set the intensity and frequency of *k*-means operation. Based on the framework, we then propose a mechanism to adaptively control the intensity and frequency of *k*-means operation during evolutionary clustering process. To evaluate the proposed framework and mechanism, a series of experiments have been carried out on both simulated and real data sets. The results show that the proposed adaptive *k*-means operation usage is able to significantly enhance the performance of evolutionary optimized data clustering.

Keywords:

Evolutionary algorithm; Clustering; Adaptive *k*-means operation

1. Introduction

Data clustering is an important task in data mining and machine learning. Cluster analysis attempts to partition a set of data objects into groups or clusters such that objects within the same cluster are similar to each other but dissimilar from objects in different clusters. Many clustering methods [1-4] have been proposed and they can be broadly divided into hierarchical and partitional clustering. For the partitional clustering, it can be further divided into hard and fuzzy clustering [5], according to whether the object is assigned to one or more clusters. Here, we confine our attention to hard partitional clustering, which considers the global shape of clusters and each object is assigned to only one cluster.

Given a data set $X = \{x_1, x_2, \ldots, x_n\}$, the task of hard partitional clustering is to partition the data set into a set of clusters $C = \{C_1, C_2, \ldots, C_k\}$ such that: 1) $C_i \neq \varnothing$, $1 \leq i \leq k$; 2) $\bigcup_{i=1}^{k} C_i = X$; 3) $C_i \bigcap C_j = \varnothing$, $i \neq j$, $1 \leq i, j \leq k$. This task is typically accomplished by optimizing certain clustering criteria [4, 6], among which the Sum of Squared Errors (SSE) is perhaps the most popular one. To deliver clustering solutions by minimizing the SSE criterion, however, is known to be a *NP*-hard problem [7]. Most methods [8, 9] use alternating optimization techniques (e.g., *k*-means algorithm), which could be highly sensitive to initialization and susceptible to local optimum.

Since evolutionary algorithms (EAs) possess good global search capability, they have been widely applied for partitional clustering by optimizing SSE. For example, Hall [10] and Maulik [11] reported the use of traditional genetic algorithm (GA) to approach the problem. Typically, for a data set with non-trivial size, traditional EAs can take a large amount of time to converge. To alleviate this issue, hybrid EAs, which incorporate *k*-means operation into traditional EAs, have been proposed in literature [12-15]. For example, Krishana [12] proposed a hybrid GA for clustering, in which the crossover operator is replaced by one iteration of *k*-means algorithm. Sheng [13], Naldi [14], and Tvrd [15], on the other hand, embed one iteration of *k*-means algorithm in EAs as the local search for data clustering. These studies show that hybridization of *k*-means operation in EAs can greatly improve their search efficiency. This is due to the *k*-means operation enhances the local search capability of EAs, therefore making the methods efficient. Furthermore, the work also shows hybrid methods can even improve the quality of solution. The reason is perhaps that the *k*-means operation is able to improve the balance of evolutionary clustering search. Although the *k*-means algorithm has been widely employed to improve the performance of EA based data clustering, it is generally applied with a fixed number of iterations (i.e., fixed intensity) and at each generation (i.e., fixed frequency) during evolution process. Since the evolution is a dynamic process, applying the *k*-means operation with fixed intensity and frequency may significantly limit their performance for data clustering.

978-1-5386-0408-3/17/$31.00 ©2017 IEEE

ALGORITHM 1. Evolutionary *K*-means Clustering

Step 1. Generate an initial population of clustering solutions.
Step 2. Evaluate the fitness for each solution in the initial population.
Step 3. Repeat the following (a) to (d) until the stopping criterion is met.
  a) Global search:
    Selection
    Crossover
    Mutation
  b) Local search: One iteration of *k*-means algorithm
  c) Fitness evaluation
  d) Generate new population
Step 4. Output the best solution from the terminal population.

In this paper, we investigate the issue of appropriate usage of *k*-means operation in evolutionary optimized data clustering. For this purpose, we first introduce a generalized *k*-means usage framework (GKUF), in which the intensity and frequency of *k*-means operation can be arbitrary set during evolutionary clustering. This is achieved by introducing two parameters *X* and *Y*, which is used to control the intensity and frequency, respectively, of *k*-means operation. Based on the framework, we then propose a mechanism to adaptively control the intensity and frequency of *k*-means. In the experiments, we first empirically show that incorporating the *k*-means operation with fixed intensity and frequency, as typically used in existing methods, is far more than optimal for data clustering. Then, experiments are carried out to evaluate the proposed adaptive *k*-means usage mechanism (AKUM). The results show that the proposed AKUM is able to significantly enhance the performance of evolutionary optimized data clustering.

2. A generalized *k*-means usage framework

In the traditional evolutionary *k*-means clustering (EKC) approach, the *k*-means operation is generally performed to fine-tune offspring solutions generated by variation operations. The pseudo-code of a typical method of this approach is shown in ALGORITHM 1. In this approach, the *k*-means operation is generally applied with a fixed intensity (typically one iteration of *k*-means algorithm) and frequency (generally applied at each generation). As the evolution is a dynamic process, employing the *k*-means operation in such a manner is not viable since different intensity and frequency of *k*-means operation may be required at different stages of evolution. To relax such a restriction, here we give a generalized *k*-means usage framework (GKUF). By incorporating the GKUF into the traditional EKC, the resulting pseudo-code is shown in ALGORITHM 2. In this framework, the intensity and frequency, controlled by parameter *Y* and *X*, respectively, can be arbitrary set during the evolution. FIGURE 1.(a) shows an instance of GKUF with *X* and *Y* value setting to be 2 and 1, respectively, representing a method applying one iteration of *k*-means algorithm for every two generations. FIGURE 1.(b) shows another instance of GKUF with dynamic values of *X* and *Y*. Clearly, the traditional manner of *k*-means usage is a special case of the GKUF with the value of *X* and *Y* both to be 1.

ALGORITHM 2. Evolutionary Clustering with a Generalized *K*-means Usage Framework

Step 1. Generate an initial population of clustering solutions, set Gen = 1.
Step 2. Evaluate the fitness for each solution in the initial population.
Step 3. Repeat the following (a) to (d) until the stopping criterion is met.
  a) Global search:
    Selection
    Crossover
    Mutation
  b) If gen % *X* = 0:
    For each offspring solution:
      Set Iter = 0
      Do:
        Run one iteration of *k*-means algorithm on the offspring solution
        Iter++
      While not arrived at local optimum or Iter < *Y*
    End For
  End If
  c) Gen++
  d) Fitness evaluation
  e) Generate a new population
Step 4. Output the best solution from the terminal population.

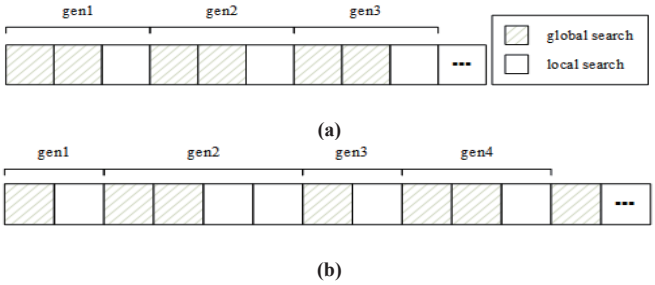


FIGURE 1. Two instances of GKUF with different *X* and *Y* values

3. Adaptive *k*-means usage

It is intuitively obvious that different intensity and frequency can be appropriate at different stages of the evolutionary process. Using rigid values of parameters *X* and *Y* in GKUF and keeping them fixed during the evolution is therefore not a viable solution. In EA literature, it has been widely agreed that maintaining appropriate population

diversity is essential for the performance of EA. Particularly, by incorporating the GKUF into an EC for data clustering, it can be observed that a too high intensity and frequency of *k*-means operation will lead to a quick loss of population diversity and trap into local optima. On the other hand, a too low intensity and frequency of *k*-means operation will result in an unnecessary large amount of time for the population of EA to converge. A potential way to appropriately use the *k*-means operation is perhaps by monitoring the population diversity during evolution. In this section, we investigate such a potential and present a mechanism, called adaptive *k*-means usage mechanism (AKUM), to adaptively control the intensity and frequency of *k*-means operation during evolutionary clustering.

Many population diversity metrics have been proposed in literature [16, 17]. Here, a metric, which is inspired by the idea presented in [18], has been employed for our purpose. In [18], the average fitness value $\bar{f}$ in relation to the maximum fitness value $f_{\max}$ of the population (i.e., $f_{\max}-\bar{f}$) is used to measure the population diversity. The idea is that the value of $f_{\max}-\bar{f}$ is likely to be smaller for a converged population than that of a population scattered in the solution space. By employing this idea, our proposed AKUM works as follows: at the initial phase of evolution, a traditional setting of intensity and frequency (i.e., X=Y=1) is used as initial values. In this phase, one iteration of *k*-means is applied to each offspring generated by global search operations. After the initial phase, the diversity of current population (i.e., $f'_{\max}-\bar{f}'$) is calculated and compared with that of previous population (i.e., $f_{\max}-\bar{f}$). If $f_{\max}-\bar{f}>f'_{\max}-\bar{f}'$, indicating that the diversity is decreased, we adjust the frequency of *k*-means operation as

$$X=\left\lfloor k_1\cdot\frac{\bar{f}'-f'_{\min}}{f'_{\max}-f'_{\min}}\right\rfloor, \tag{1}$$

while keeping the intensity to be 1. Here, $f'_{\max}$ and $f'_{\min}$ are the maximum and minimum fitness values, respectively, of current population, and $\bar{f}'$ is the average fitness. The term $f'_{\max}-f'_{\min}$ is used to normalize the value of $\bar{f}'-f'_{\min}$ and k_1 is a scaling constant. According to the above equation, a lower frequency of *k*-means operation will therefore be employed when the population tends to converge. On the other hand, if $f_{\max}-\bar{f}\leqslant f'_{\max}-\bar{f}'$, we adjust the intensity of *k*-means operation as

$$Y=\left\lfloor k_2\cdot\frac{f'_{\max}-\bar{f}'}{f'_{\max}-f'_{\min}}\right\rfloor, \tag{2}$$

and keep the frequency of *k*-means operation to be 1. Here, k_2 is a scaling constant. Such an equation means a higher intensity of *k*-means operation will be employed for the population with a larger diversity. The core idea of the above mechanism is to strength the local search when the diversity of population tends to increase, and vice versa.

4. AKUM based genetic clustering algorithm

As a general mechanism, the proposed AKUM can be embedded into any evolutionary algorithm for clustering by optimizing the SSE criterion. Here, for evaluation purpose we incorporate the AKUM into a simple genetic algorithm, denoted as AKUM_SGA. In this section, we describe the setting of AKUM_SGA, including solution representation, genetic operation and fitness evaluation.

For representation, we adopt the real value representation, which is commonly used in most of the recent EA based clustering methods [19]. Here, real values are used to represent cluster centers. In this representation, for a *d*-dimension data set which has *k* clusters, a real vector of $k\times d$ is used to represent the solution. The *d*-dimensional values of the first cluster is located at the first *d* positions in the chromosome, the center of the second cluster is at the next *d* positions, and so on.

For crossover operation, which is used to recombine the parents to produce offspring, we employ the arithmetic crossover scheme [20]. The crossover is performed on each pair of parents with probability of P_c. After crossover operation, a low probability P_m of Gaussian mutation [21] is then applied to the individual solution. For selection operation, the roulette wheel strategy [21] is adopted. This operator randomly selects an individual solution from the previous population according to the distribution of fitness values of solutions.

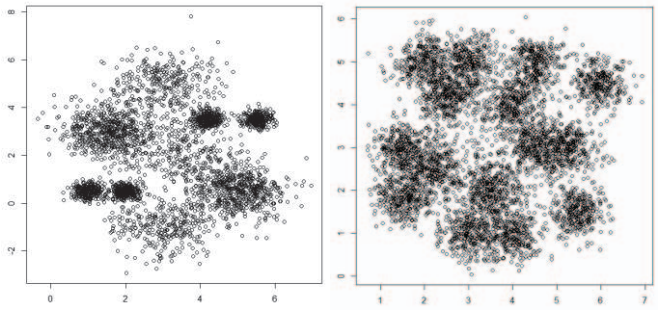
The fitness of individual solution is used to measure its goodness, which is calculated based on the SSE criterion. Specifically, the SSE is defined as:

$$SSE=\sum_{i=1}^{n}\sum_{j=1,x_i\in C_j}^{k}\left|x_i-m_j\right|^2, \tag{3}$$

where

$$m_j=\frac{1}{\left|C_j\right|}\sum_{x\in C_j}x\cdot \tag{4}$$

After obtaining the SSE, the fitness is thus defined as f = 1/*SSE*, so that maximizing the fitness is to minimize the SSE value. It is possible that certain partitions may result in empty clusters. To punish such clustering solution, the fitness function is further defined as f=1/*SSE*′ and *SSE*′=*SSE*+(*b*/*s*)**SSE*, where *s* is the total number of clusters and *b* is the number of empty clusters.



(a) *Normal*_9 **(b)** *Normal*_15

FIGURE 2. Simulated data sets

5. Experiments

In this section, after describing the data sets in subsection 5.1, we present parameter settings of the algorithms used in experiments in subsection 5.2. We first examine the GKUF with different parameter configurations on different clustering problems. After that, we evaluate the performance of the proposed mechanism. All results reported in this section were obtained on a PC with an Intel^{R} Core^{TM} i5-2400 CPU @ 3.10GHz running Windows7 operation system. Unless otherwise stated, the results are averaged over 100 runs of algorithms.

5.1. Data sets

Both simulated and real data sets have been used in our experiments. The simulated data sets, as shown in FIGURE 2, are generated according to a spherical bivariate normal distribution, representing different degrees of difficulty for clustering. The first data set, *Normal*_9, appears to be relatively simple: one highly sparse cluster in the middle surrounded by four dense clusters as well as four sparse clusters. For the second data, *Normal*_15, the clusters have rather different sizes and volumes, and many of them overlapping each other. In order to increase the difficulty of problem solving further, many noise data points have also been added in both data sets.

For real data sets, we first consider the *Iris* and *Breast cancer*, which are from UCI Machine Learning Repository [22]. The *Iris* data set contains 3 classes of 50 data objects each, where each class refers to a type of iris plant (i.e., Setosa, Versicolor, and Virginica). The *Breast cancer* data set has 683 data objects with two clusters. Each data object has 9 variables which correspond to clump thickness, cell size uniformity, cell shape uniformity, marginal adhesion, single epithelial cell size, bare nuclei, bland chromatin, normal nucleoli, and mitoses.

In addition, we also consider the gene expression data, specifically, the *Subcellcycle* and *Yeast*2945 have been used in experiments. They are subsets of the yeast cell cycle data. The *Subcellcycle*, provided by Cho in [23], consists of 384 genes whose expression levels peak at different time-points corresponding to the five phases (i.e. early G1, late G1, S, G2, and M) of the cell cycle. The *Yeast*2945 is provided by Tavazoie in [24], in which 2945 genes are selected out of 6220. Both data sets are standardized by *Z*-score so that every attribute has an average expression value of zero and a standard deviation equal to one. A list of data sets used in the experiments is shown in TABLE 1.

TABLE 1. A list of data sets used in the experiments

Data sets	*No. of data points*	*No. of attributes*	*No. of clusters*
*Normal*_9	3300	2	9
*Normal*_15	4800	2	15
Iris [22]	150	4	3
Breast cancer [22]	683	9	2
Subcellcycle [23]	387	17	5
*Yeast*2945 [24]	2945	15	30

TABLE 2. The number of cases that the GKUF with different combinations of *X* and *Y* values outperform the GKUF(1, 1) in terms of SSE and/or running time

Data sets	*SSE values*	*Running time*	*SSE and running time*
Iris	192	340	145
Subcellcycle	219	247	145
Breast cancer	372	206	199
*Yeast*2945	350	76	29

5.2. Parameter settings

For experiments, several parameters of GKUF and AKUM_SGA need to be set. These include the population size, crossover rate, mutation rate, stopping criteria and scaling constants k_1 and k_2 in AKUM. These parameters are set as follows. The population size is set to be 30, and the crossover and mutation rate is 0.7 and 0.03, respectively. We terminate the algorithm when the best individual in population has not changed for 20 generations. The value of scaling constants k_1 and k_2 in AKUM_SGA are set to be 8 and 5, respectively.

TABLE 3. The performance of GCA and AKUM_SGA on experimental data sets

Data sets	GCA		AKUM_SGA	
	SSE values	*Running time (Second)*	*SSE values*	*Running time (Second)*
Normal_9	527.78	0.757	527.46	0.459
Normal_15	442.82	1.54	442.72	1.12
Iris	140.97	0.0123	140.97	0.0089
Breast cancer	2728.14	0.0669	2728.14	0.0488
Subcellcycle	1943.60	0.226	1942.64	0.185
*Yeast*2945	20332.34	12.00	20308.20	10.27

TABLE 4. The performance of GKA and AKUM_GKA on experimental data sets

Data sets	GKA [12]		AKUM_GKA	
	SSE values	*Running time (Second)*	*SSE values*	*Running time (Second)*
Normal_9	532.78	8.29	527.85	5.36
Normal_15	446.49	19.20	442.70	13.50
Iris	140.97	0.0241	140.97	0.0237
Breast cancer	2728.41	0.366	2728.42	0.379
Subcellcycle	1965.20	0.548	1964.04	0.298
*Yeast*2945	20291.25	68.79	20286.00	68.51

5.3. Results

We first empirically test the GKUF using different combination of X and Y values, and compare the results with the traditional k-means usage (i.e., one iteration of k-means operation is employed at each generation) on the experimental data sets. Due to a large number of combinations with different values of X and Y exist, we consider value of X and Y up to 20 (i.e., $1 \leqslant X \leqslant 20$, $1 \leqslant Y \leqslant 20$). For convenience, the term GKUF(X, Y) is used to denote the GKUF with a specific parameter value of X and Y. The GKUF(1, 1) represents an algorithm with the traditional k-means usage. We report the number of cases that the GKUF with different combinations of X and Y values outperform the GKUF(1, 1) in terms of SSE and/or running time. The results are shown in TABLE 2. It is can be seen from the results that there are many cases of the GKUF with different values of X and Y can outperform the GKUF(1, 1). Particularly, TABLE 2 shows that, for most of the data sets, there exist many different combinations of X and Y values, in which GKUF outperforms GKUF(1, 1) in terms of both SSE value and efficiency. For example, on the *Iris* data, GKUF can be more effective and efficient than GKUF(1, 1) in 145 cases out of 400. Thus, it could be concluded that traditional manner of k-means usage is far more than optimal for data clustering.

Next, we carry out experiments to evaluate the performance of adaptive k-means usage mechanism, which is used to dynamically set the values of X and Y. For this purpose, we carry out two pairwise comparisons. In the first pairwise comparison, we compare the AKUM_SGA with its standard version (i.e., the traditional manner of k-means usage is employed rather than the AKUM, denoted as GCA) based on the same settings. In the second pairwise comparison, we incorporate our proposed AKUM into the genetic k-means clustering algorithm (GKA) [12], resulting a method denoted as AKUM_GKA, and compare its performance with the original GKA. The same parameter settings, as specified in [12], are used for GKA and AKUM_GKA. While the k_1 and k_2 in AKUM_GKA is taken to be 3 and 5, respectively. We compare their performance in terms solution quality and computational efficiency. Therefore, we report average SSE values and average running time. The results of two pairwise comparisons are shown in TABLE 3 and TABLE 4, respectively.

The results in TABLE 3 clearly shows that, in comparison with the GCA, the AKUM_SGA is able to achieve better solution quality in terms of SSE values on all data sets. It can be noticed that the performance improvement of AKUM_SGA can be particularly significant on the data sets with large size and complex search space. For example, for the GCA on *Yeast*2945 data, the average SSE value of solutions turns out to be 20332.34. By comparison, the AKUM_SGA gives an average SSE value of 20308.20. Further, the AKUM_SGA can be significantly faster than the GCA. For example, on *Breast cancer* data, the GCA needs 0.0669 seconds, while the AKUM_SGA takes only 0.0488 seconds on average. The performance improvement of AKUM_SGA over the GCA is solely due to the dynamically usage of k-means operation introduced by the proposed mechanism.

The results of second pairwise comparison, between the GKA and AKUM_GKA, are shown in TABLE 4. From the table we can see that the SSE values delivered by the AKUM_GKA are better than those of GKA on most data sets, except *Breast cancer* data. For example, on *Yeast*2945 data, by average, the GKA achieves solutions with SSE value of 20291.25. By contrast, for the AKUM_GKA, it provides solutions with an average SSE of 20286.00. The comparable performance of the GKA and AKUM_GKA on the *Breast*

cancer data is due to the search space of the problem is rather simple, while our proposed mechanism is particularly suitable for problem with large and complex search spaces. Moreover, in terms of running time, the results show that for most of data sets, except the *Breast cancer* data, the AKUM_GKA can be much efficient than GKA. For example, on *Subcellcycle* data, the AKUM_GKA is nearly 45% faster than the GKA.

6. Conclusions

This paper investigates the issue of appropriate usage of *k*-means operation in evolutionary optimized data clustering. We first introduce a generalized *k*-means usage framework, in which the intensity and frequency of *k*-means operation can be arbitrary set during evolution. Based on the framework, a mechanism is then proposed to adaptively control the intensity and frequency of *k*-means operation during evolutionary process. For evaluation purpose, the proposed mechanism is incorporated into a simple GA for clustering. In the experiments, we show that traditional manner of *k*-means operation usage is far more than optimal. More importantly, we show that the proposed adaptive *k*-means operation usage is able to significantly enhance the performance of evolutionary optimized data clustering.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant No. 61573316).

References

[1] Tseng, V. S., and C. P. Kao, "Efficiently mining gene expression data via a novel parameterless clustering method", IEEE/ACM Transactions on Computational Biology & Bioinformatics 2.4(2005):355-365.

[2] Lee, Jaewook, and D. Lee, "Dynamic Characterization of Cluster Structures for Robust and Inductive Support Vector Clustering", IEEE Transactions on Pattern Analysis & Machine Intelligence 28.28(2006):1869-1874.

[3] R. Vidal, "Subspace Clustering", IEEE Signal Processing Magazine 28.2(2011):52-68.

[4] Anil K. Jain, "Data Clustering: 50 Years Beyond K-means", Pattern Recognition Letters 31.8(2010):651-666.

[5] Bezdek J.C. and N.R. Pal, "Some New Indexes of Cluster Validity", IEEE Trans. on Systems, Man, and Cybernetics-Part B: Cybernetics, vol.28, no.3, 1998.

[6] R.O. Duda, P.E. Hart and D.G. Stork, "Pattern classification", 2nd edn., Wiley, New York, 2001.

[7] Hruschka, Eduardo Raul, R. J. G. B. Campello, and A. A. Freitas, "A Survey of Evolutionary Algorithms for Clustering", IEEE Transactions on Systems Man & Cybernetics Part C 39.2(2009):133-155.

[8] Macqueen, J, "Some Methods for Classification and Analysis of MultiVariate Observations", Proc. of, Berkeley Symposium on Mathematical Statistics and Probability 1967:281-297.

[9] Chen, Wei‐Chen, and R. Maitra, "Model‐based clustering of regression time series data via APECM—an AECM algorithm sung to an even faster beat", Statistical Analysis & Data Mining 4.6(2011):567-578.

[10] Hall L.O., I.B. Ozyurt and J.C. Bezdek, "Clustering with a genetically optimized approach", IEEE Trans. on Evolution Computation, vol.3, no.2, pp.103-112, 1999.

[11] Maulik U. and S. Bandyopadhyay, "Genetic- Algorithm-Based Clustering Technique", Pattern Recognition, vol.33, pp.1455-1465, 2000.

[12] K. Krishna, M.N. Murty, "Genetic k-means algorithm," IEEE Transactions on Systems, Man and Cybernetics, Part B, Cybernetics, 29(3), pp.433-439, 1999.

[13] Sheng Weiguo, et al, "Multilocal Search and Adaptive Niching Based Memetic Algorithm With a Consensus Criterion for Data Clustering", IEEE Transactions on Evolutionary Computation 18.5(2014):721-741.

[14] Naldi, M. C, and R. J. G. B. Campello, "Evolutionary k-means for distributed data sets", Neurocomputing 127.3 (2014): 30-42.

[15] Tvrd, et al, "Hybrid differential evolution algorithm for optimal clustering", Applied Soft Computing 35.C (2015): 502-512.

[16] Shen, Xiaoning, M. Zhang, and T. Li, "A Multi-objective Optimization Evolutionary Algorithm Addressing Diversity Maintenance", International Joint Conference on Computational Sciences and Optimization IEEE Computer Society, 2009:524-527.

[17] X Li, J. Branke, and M. Kirley. Performance Measures and Particle Swarm Methods for Dynamic Multiobjective Optimization Problems. 2007 Genetic and Evolutionary Computation Conference, D. Thierens, Ed., vol. 1. London, UK: ACM Press, July 2007, p. 90.

[18] Srinivas, M, and L. M. Patnaik, "Adaptive probabilities of crossover and mutation in genetic algorithms", IEEE Transactions on Systems Man & Cybernetics 24.4 (1994): 656-667.

[19] Hruschka E.R., R. Campello, A. Freitas and A.C.P.L.F. de Carvalho, "A survey of evolutionary algorithms for clustering", IEEE Transactions on System, Man, and Cybernetics: Part C, vol.39, no.2, pp.133-55, 2009.

[20] Michalewicz Z., "Genetic Algorithms + Data Structure = Evolution Programs", Third ed. New York: Springer-Verlag, 1996.

[21] T. Back, "Evolutionary algorithms in theory and practice", Oxford University Press, 1996

[22] UCI Machine Learning Repository, Retrieved from http:// http://archive.ics.uci.edu/ml/.

[23] Cho, R. J., et al, "A genome-wide transcriptional analysis of the mitotic cell cycle", Molecular Cell 2.1 (1998): 65.

[24] Tavazoie, S, et al, "Systematic determination of genetic network architecture", Nature Genetics 22.3(1999):281.