Differential Evolution-based Weighted Combination of Distance Metrics for *k*-means Clustering

Muhammad Marwan Muhammad Fuad

Forskningsparken 3, Institutt for kjemi, NorStruct
The University of Tromsø - The Arctic University of Norway
NO-9037 Tromsø, Norway
marwan.fuad@uit.no

Abstract: Bio-inspired optimization algorithms have been successfully used to solve many problems in engineering, science, and economics. In computer science bio-inspired optimization has different applications in different domains such as software engineering, networks, data mining, and many others. One of the main tasks in data mining is clustering, namely *k*-means clustering. Distance metrics are at the heart of all data mining tasks. In this paper we present a new method which applies differential evolution, one of the main bio-inspired optimization algorithms, on a time series *k*-means clustering task to set the weights of the distance metrics used in a combination that is used to cluster the time series. The weights are obtained by applying an optimization process that gives optimal clustering quality. We show through extensive experiments how this optimized combination outperforms all the other stand-alone distance metrics, all by keeping the same low complexity of the distance metrics used in the combination.

Keywords: Evolutionary Computing, Differential Evolution, Distance Metrics, *k*-means Clustering, Time Series Data Mining.

1 Introduction

Global optimization is a ubiquitous problem that has a very broad range of applications in engineering, economics, and others. In computer science optimization has different applications in software engineering, networking, data mining and other domains. Optimization can be defined as the action of finding the best-suited solution of a problem within given constraints. These constraints can be in the boundaries of the parameters controlling the optimization problem, or in the function to be optimized. Optimization problems can be classified according to whether they are: discrete/continuous/hybrid,constrained/unconstrained, single objective/multiobjective, unimodal (one extreme point) /multimodal (several extreme points).

Formally, an optimization task can be defined as follows: Let $\vec{X} = [x_1, x_2, ..., x_{nbp}]$ be the candidate solution to the problem for which we are searching an optimal

solution. Given a function $f : U \subseteq \mathsf{R}^{nbp} \rightarrow \mathsf{R}$, find the solution $\overrightarrow{X^*} = \left[x_1^*, x_2^*, ..., x_{nbp}^*\right]$ (*nbp* is the number of parameters) which satisfies $f\left(\overrightarrow{X^*}\right) \leq f\left(\overrightarrow{X}\right), \forall \overrightarrow{X} \in U$. The function *f* is called the *fitness function*, the *objective function*, or the *cost function*. It is worth mentioning here that it is a convention for optimization problems to be expressed as minimization problems since any maximization optimization problem can be transformed into a minimization problem.

Metaheuristics are probabilistic optimization algorithms which are applicable to a large variety of optimization problems. Many of these metaheuristics are inspired by natural processes, natural phenomena, or by the collective intelligence of natural agents, hence the term *bio-inspired*, also called *nature-inspired*, optimization algorithms.

Bio-inspired optimization can be classified into two main families; the first is *Evolutionary Algorithms* (EA). This family is probably the largest family of bio-inspired algorithms. EA are population based algorithms that use the mechanisms of Darwinian evolution such as selection, crossover and mutation. Of this family we cite *Genetic Algorithms* (GA), *Genetic Programming* (GP), *Evolution Strategies* (ES), and *Differential Evolution* (DE). The other family is *Swarm Intelligence* (SI). This family uses algorithms which simulate the behavior of an intelligent biological system. Of this family we mention *Particle Swarm Intelligence* (PSO), *Ant Colony Optimization* (ACO), and *Artificial Bee Colony* (ABC). Fig. 1 shows the main bio-inspired metaheuristics.

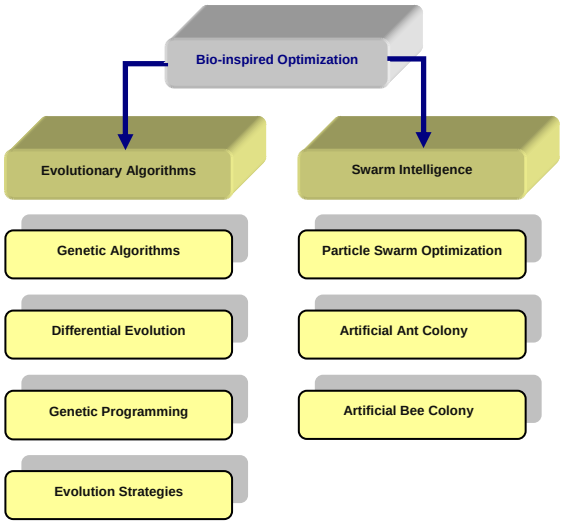

Fig. 1. Some of the main bio-inspired metaheuristics

Data mining is a field of computer science which handles several tasks such as classification, clustering, anomaly detection, and others. Processing these tasks usually requires extensive computing. As with other fields of computer science, different papers have proposed applying bio-inspired optimization to data mining tasks [11], [12], [13], [14], [15].

In this paper we apply one bio-inspired optimization technique on a particular task of time series data mining which is *k*-means clustering. This task includes using a distance metric or a similarity measure. In this work we use a weighted combination of distance metrics to cluster the time series. The novelty of our work is that the weights of the combination are obtained through an optimization process using differential evolution as an optimizer. The experiments we conducted clearly show how the proposed combination can enhance the quality of the *k*-means clustering of time series compared with the clustering quality obtained when using the distance metrics that constitute the combination as stand-alone distances.

The rest of the paper is organized as follows; the related work is presented in Section 2, in Section 3 we introduce the new algorithm, which we test in Section 4. We conclude this paper with Section 5.

2 Related Work

A *time series* is an ordered collection of observations at intervals of time points. These observations are real-valued measurements of a particular phenomenon.

Time series data mining handles several tasks such as classification, clustering, similarity search, motif discovery, anomaly detection, and others.

Clustering, also called *unsupervised learning*, is partitioning of the data objects into groups, or clusters, so that the objects within a cluster are similar to one another and dissimilar to objects in other clusters. [8]. There are several basic clustering methods such as *Partitioning Methods*, *Hierarchical Methods*, *Density-Based Methods*, and *Grid-Based Methods*. *k-means*, is a centroid-

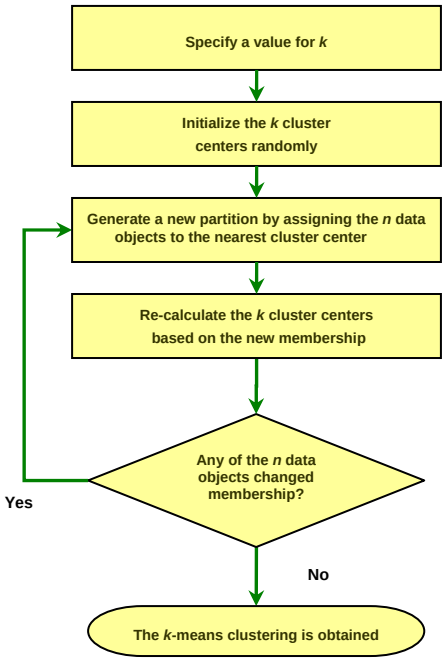

Fig. 2. Flow chart of the *k*-means clustering algorithm

based partitioning technique which uses the *centroid* (also called *center*) of a cluster; c_i , to represent that cluster. Conceptually, the centroid of a cluster is its center point. The centroid can be defined in various ways such as by the mean of the objects assigned to the cluster. *k*-means is one of the most widely used and studied clustering formulations [9] . In *k*-means clustering we are given a set of *n* data points in *d*-dimensional space R^d and an integer *k* and the problem is to determine a set of *k* points in R^d, the centroids, so as to minimize the mean distance from each data point to its nearest center [9]. More formally, the *k*-means clustering error can be measured by:

$$E = \sum_{i=1}^{k} \sum_{j=1}^{n_j} d\left(u_{ij}, c_i\right) \quad \textbf{(1)}$$

Where u_{ij} is the *i*th point in the *j*th cluster, and n_j is the number of points in that cluster. The quality of the *k*-means clustering increases as the error given in relation (1) decrease. Fig. 2 shows the flow chart of the *k*-means algorithm.

The number of clusters is decided by the user, or application-dependent, or given by some cluster validity measure.

The *k*-means starts by selecting the centroids c_i , which are usually chosen randomly. In step two the membership of each of the *n* data points is determined by assigning it to the nearest cluster centroid. In step three c_i are re-calculated assuming the memberships obtained in step two are correct. If none of the *n* data objects have changed its membership the algorithm stops otherwise it goes back to step tow. Fig. 3 shows an example of the different steps of the *k*-means clustering with n=30 and k=3.

The concept of similarity on which clustering, and other data mining tasks, is based is a fundamental one in data mining. In the similarity search problem a pattern or a query is given and the similarity search algorithm seeks to retrieve the data objects in the database that are "close" to that query according to some semantics that quantify this closeness. This closeness or similarity is quantified using a principal concept which is the *similarity measure* or its strongest form; the *distance metric*. Distance metrics satisfy the well-known metric axioms (non-negativity, symmetry, identity, triangle inequality). Metric spaces have many advantages, the most famous of which is that a single indexing structure can be applied to several kinds of queries and data types that are so different in nature. This is mainly important in establishing unifying models for the search problem that are independent of the data type. This makes metric spaces a solid structure that is able to deal with several data types [16]

There are many similarity measures and distance metrics that are widely used in the field of time series data mining; the most-widely known is the *Minkowski distance*. This is actually a whole family of distances, the most famous of which are:

i- Euclidean Distance (L_2)- defined between time series *S* and *T* as:

$$L_2(S,T) = \sqrt{\sum_{i=1}^{n} (s_i - t_i)^2} \quad \textbf{(2)}$$

ii- Manhattan Distance (L_1)- defined as:

$$L_1(S,T) = \left| \sum_{i=1}^{n} (s_i - t_i) \right| \qquad (3)$$

This distance is also called the *city block distance.*

iii- Maximum Distance (L_∞)- defined as:

$$L_\infty(S,T) = \max_i (|s_i - t_i|) \qquad (4)$$

This distance is also called the *infinity distance* or the *chessboard distance*. Fig. 4 shows a few examples of the Minkowski distance.

It is important to mention here that one of the advantages of the members of the Minkowski distance is their low computational complexity which is $O(n)$. It is also important to emphasize, and this is related to the experimental section of our paper, that the aforementioned distances are all distance metrics. A last note about this, which is also related to the experimental section of this work, is that all these distances are applicable only to time series of the same length.

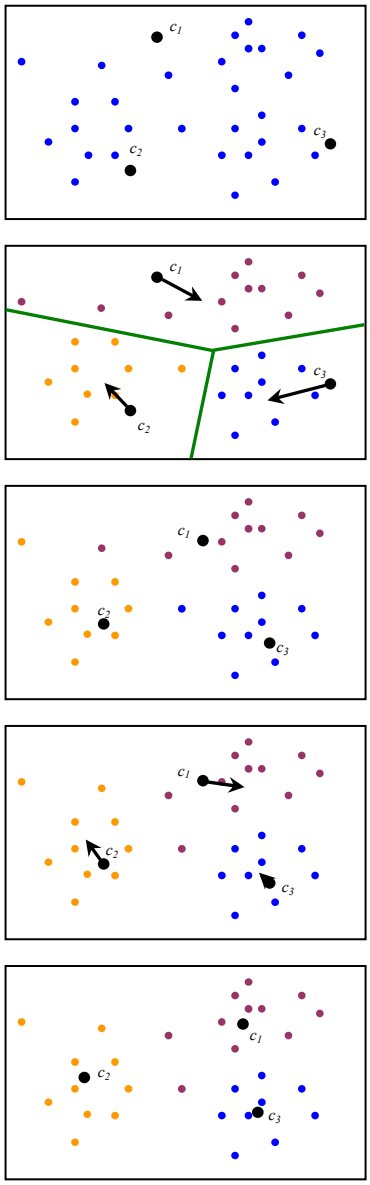

Fig. 3. The different steps of the *k*-means clustering algorithm

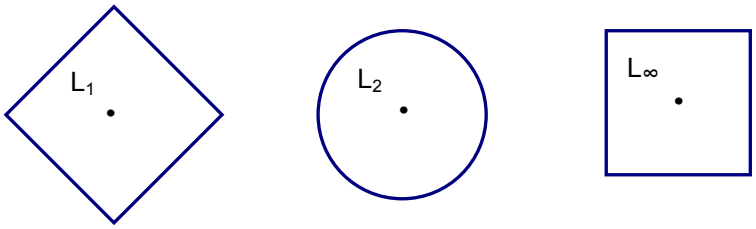

Fig. 4. From left to right; the Manhattan distance, the Euclidean distance, and the infinity distance

3 Using a Combination of Distance Metrics for *k*-means Clustering

Instead of using one similarity measure or distance metric to handle data mining tasks, we can use a combination of several similarity measures or distance metrics to get better results. This idea has been proposed by several researchers before. In [2] the authors propose utilizing a similarity function defined as a weighted combination of several metrics to handle the similarity search problem. A similar idea was proposed in [3] where the authors present a retrieval method based on a weighted combination of feature vectors. However, these two works do not suggest using any optimization algorithm to determine the weights.

In this paper we propose utilizing a weighted combination of distance metrics to handle the *k*-means clustering task of time series data. The novelty of our method is: (i) the weights are determined as the outcome of an optimization process and (ii) it proposes a combination of distance metrics to handle a clustering task of time series data.

Formally, we perform a *k*-means clustering task of time using a combination d which is defined as:

$$d(S,T) = \sum_{i=1}^{n} \omega_i d_i(S,T) \tag{5}$$

where $\omega_i \in [0,1]$.

Notice that we could also impose that $\sum_i \omega_i = 1$, but this would not make any difference as this latter condition is simply a normalized version of the one used in (5).

As mentioned earlier, we determine the weights ω_i through an optimization process, where the objective function to be maximized is the quality of the *k*-means clustering. The optimization algorithm we use is differential evolution.

3.1 Differential Evolution

Differential Evolution (DE) is an optimization method based on the principles of genetics and natural selection. DE is considered as one the most powerful stochastic optimization algorithms for continuous parameters [4]. DE has the same elements as a standard evolutionary algorithm; i.e. a population of individuals, selection according to fitness, crossover, and random mutation. DE creates an environment in which a population of individuals, representing solutions to a particular problem, is allowed to evolve under certain rules towards a state that minimizes the value of the fitness function.

As with other evolutionary algorithms, the first step of DE is defining the problem variables and the fitness function. The range of the variable values can be constrained or unconstrained. A particular configuration of variables produces a certain value of the fitness function and the objective of DE is to find the configuration that gives the optimal value of the fitness function.

DE has many variations, but in the following we present the classical DE. DE starts with a collection of randomly chosen individuals constituting a population, whose size is *popsize*. Each of these solutions is a vector of *nbp* dimensions and it represents a possible solution to the problem at hand. The fitness function of each individual is evaluated. The next step is optimization. In this step for each individual of the population, which we call the *target vector* $\vec{T}_i$ at this stage, three mutually distinct individuals $\vec{V}_{r1}, \vec{V}_{r2}, \vec{V}_{r3}$, and different from $\vec{T}_i$, are chosen at random from the population (hence the minimum value of *popsize* is 4). The *donor vector* $\vec{D}$ is formed as a weighted difference of two of $\vec{V}_{r1}, \vec{V}_{r2}, \vec{V}_{r3}$, added to the third; i.e. $\vec{D} = \vec{V}_{r1} + F\left(\vec{V}_{r2} - \vec{V}_{r3}\right)$. F is called the *mutation factor* or the *differentiation constant* and it is one of the *control parameters* of DE. F is usually chosen from $[0,1[$.

The *trial vector* $\vec{R}$ is formed from elements of the target vector $\vec{T}_i$ and elements of the donor vector $\vec{D}$ according to different schemes such as the *exponential* and the *binomial* ones [1]. In the following we present the crossover scheme presented in [6] which we adopt in this paper; an integer *Rnd* is randomly chosen among the dimensions $[1, nbp]$. This guarantees that at least one of the dimensions will be changed. Then the trial vector $\vec{R}$ is formed as follows:

$$t_i = \begin{cases} t_{i,r1} + F\left(t_{i,r2} - t_{i,r3}\right) & \text{if } \left(rand_{i,j}[0,1[< C_r\right) \vee \left(Rnd = i\right) \\ t_{i,j} & \text{otherwise} \end{cases} \quad \textbf{(6)}$$

where $i = 1,...,nbp$. C_r is the *crossover constant*, which is another control parameter. The control parameters of DE are determined by the algorithm designer.

The next step of DE is selection. This step decides which of the trial vector and the target vector will survive in the next generation and which will die out. The selection

is based on which of the two vectors; trial and target, yields a better value of the fitness function.

Crossover and selection repeat for a certain number of generations *NrGen* , which is the third control parameter of DE. Most algorithms add a *stopping criterion,* which terminates DE if met, even if *NrGen* has not been reached.

4 Experiments

We conducted extensive experiments using time series datasets of different sizes and dimensions available at UCR [10]. This archive makes up between 90% and 100% of all publicly available, labeled time series datasets in the world, and it represents the interest of the data mining/database community, and not just one group [5].

The distances we are using in the combination in relation (5) are the Euclidean distance, the Manhattan distance, and the maximum distance (relations (2), (3), and (4)).

In the time series data mining community *Dynamic Time Warping* (DTW) [7] is widely used, however we decided not to include it in the combination for several reasons; first, DTW is a similarity measure and not a distance metric, while L_2, L_1, L_∞ are all distance metrics, so their combination will result in a distance metric, while combining a similarity measure (such as DTW) with distance metrics will result in a similarity measure (this can be easily proved mathematically). The second reason why we are not adding DTW to the combination is that DTW has a higher complexity, which is $O(mn)$ (or $O(n^2)$ if the two time series have the same length), whereas, the three other distances have a complexity, as mentioned in Section 2, of $O(n)$. The third reason for not adding DTW is that it is applied to time series of different lengths, which is not the case with the other three distances. For all these reasons we decided to exclude DTW from the combination despite its widespread use in time series data mining, so our final combination is:

$$d(S,T) = \omega_1 L_1(S,T) + \omega_2 L_2(S,T) + \omega_3 L_\infty(S,T) \quad (7)$$

We tested our method on a variety of datasets; the length of the time series varied between 60 (Synthetic_control) and 1639 (CinC_ECG_torso). The size of the training sets varied between 20 (SonyAIBORobot Surface) and 467 (ChlorineConcentration). The size of the testing sets varied between 30 (OliveOil) and 3840 (ChlorineConcentration), so as we can see, we tested our method on a wide range of datasets of different lengths and sizes to avoid getting biased results.

For each dataset the experiment consists of two phases; the training phase and the testing phase. In the training phase we perform an optimization process where the parameters of the optimization problem are the weights $\omega_i ; i \in [1,3]$. The objective function is the *k*-means clustering quality which we seek to maximize. The outcome of this optimization problem is the weights ω_i which give the optimal *k*-means clustering quality (c.f. Section 2).

In the testing phase, these optimal weights are used on the corresponding testing datasets to evaluate the quality of the *k*-means clustering.

As for the elements of the DE, we used the following : the population size *popsize* was 12, the number of generations *NrGen* was set to 100, the differentiation constant F was set to 0.9, and the crossover constant C_r was set to 0.5. The dimension of the problem *nbp*, as we mentioned earlier, is ω_i. Table 1 summarizes the values of the control parameters of DE used in the experiments.

Table 1. The values of the control parameters of DE used in the experiments

popsize	Population size	12
NrGen	Number of generations	100
F	Differentiation constant	0.9
C_r	Crossover constant	0.5
nbp	Number of parameters	3

Table 2 shows the optimal weights for the three distances metric for the different training datasets after running the algorithm for 100 generations. As we can see the weights vary between 0 and 0.97, which proves that some distance metrics are more effective for clustering certain datasets than others.

In the next phase we use these weights shown in Table 2 on the corresponding testing datasets to get the *k*-means clustering quality. Table 3 shows the *k*-means clustering quality for the combination together with those for L_2, L_1, L_∞ for comparison.

Table 3 shows that the clustering quality of the combination of the three distance metrics for all the datasets outperforms that of all the other three distance metrics for L_2, L_1, L_∞ as stand-alone distance metrics, which proves the validity of our proposed algorithm.

Table 2. Weights assigned to each distance metric after 100 generations on the training datasets

dataset	ω_1	ω_2	ω_3
Synthetic_control	0.34	0.39	0.76
OSULeaf	0.95	0.94	0.25
Lighting2	0.57	0.09	0.97
Lighting7	0.86	0.21	0.27
SonyAIBORobotSurfac	0.55	0.80	0.22
FaceFour	0.40	0.16	0.58
ECG200	0.01	0.02	0.93
Yoga	0.34	0.88	0.11
OliveOil	0.72	0.83	0.87
CinC_ECG_torso	0.43	0.20	0.38
ChlorineConcentration	0.00	0.00	0.67
Haptics	0.63	0.51	0.26
MedicalImages	0.63	0.11	0.69
Cricket_X	0.02	0.01	0.88
Cricket_Y	0.36	0.12	0.76

Table 3. The *k*-means clustering quality of the combination and L_2, L_1, L_∞ on the testing datasets

dataset	*k*-means clustering quality			
	L_1	L_2	L_∞	combination
Synthetic_control	0.57	0.71	0.64	0.73
OSULeaf	0.39	0.40	0.33	0.41
Lighting2	0.56	0.63	0.63	0.65
Lighting7	0.54	0.57	0.50	0.64
SonyAIBORobotSurfac	0.87	0.66	0.69	0.92
FaceFour	0.61	0.54	0.55	0.67
ECG200	0.69	0.69	0.62	0.72
Yoga	0.50	0.48	0.48	0.51
OliveOil	0.57	0.57	0.57	0.58
CinC_ECG_torso	0.49	0.47	0.46	0.52
ChlorineConcentration	0.40	0.40	0.40	0.41
Haptics	0.33	0.32	0.32	0.34
MedicalImages	0.33	0.34	0.30	0.37
Cricket_X	0.30	0.27	0.30	0.31
Cricket_Y	0.31	0.32	0.36	0.38

5 Conclusion

In this paper we presented a new algorithm for *k*-means clustering of time series data using a combination of weighted distance metrics. The weights of the combination are obtained through an optimization process where the optimizer is differential evolution; one of the most effective bio-inspired optimization algorithms for continuous optimization problems, all by keeping a low complexity of the combination. The extensive experiments we conducted show the superiority of our proposed combination over other, widely-used distance metrics, as stand-alone distance metrics.

As future work, we like to study how our proposed algorithm can be extended to cluster streaming data as an important application in time series data mining.

References

1. Biswas, A., Dasgupta, S., Das, S., and Abraham, A.: A Synergy of Differential Evolution And Bacterial Foraging Algorithm for Global Optimization. Neural Netw. World, vol. 17, no. 6, pp. 607–626, (2007).
2. Bustos, B. and Skopal, T.: Dynamic Similarity Search in Multi-metric Spaces. In Proceedings of the ACM Multimedia, MIR Workshop. ACM Press, New York, NY, 137–146 (2006).

3. Bustos, B., Keim, D. A., Saupe, D., Schreck, T., and Vrani´c, D.: Automatic Selection and Combination of Descriptors for Effective 3D Similarity Search. In Proceedings of the IEEE International Workshop on Multimedia Content-based Analysis and Retrieval. IEEE Computer Society, 514–521 (2004).
4. Das, S., and Suganthan, P.N.: Differential Evolution: A Survey of the State-of-the-Art. IEEE Trans. on Evolutionary Computation, Feb. (2011)
5. Ding, H., Trajcevski, G., Scheuermann, P., Wang, X., and Keogh, E.: Querying and Mining of Time Series Data: Experimental Comparison of Representations and Distance Measures. In Proc of the 34th VLDB (2008).
6. Feoktistov, V., Differential Evolution: In Search of Solutions (Springer Optimization and Its Applications). Secaucus, NJ, USA: Springer- Verlag New York, Inc., (2006).
7. Guo, AY., and Siegelmann, H.: Time-warped Longest Common Subsequence Algorithm for Music Retrieval, in Proc. ISMIR (2004).
8. Han, J., Kamber, M., Pei, J.: Data Mining: Concepts and Techniques, 3rd edn. Morgan Kaufmann (2011)
9. Kanungo, T., Netanyahu, N.S., Wu, A.Y.: An Efficient K-means Clustering Algorithm: Analysis and Implementation. IEEE Transactions on Pattern Analysis and Machine Intelligence 24(7) (2002)
10. Keogh, E., Zhu, Q., Hu, B., Hao. Y., Xi, X., Wei, L. and Ratanamahatana, C.A.: The UCR Time Series Classification/Clustering Homepage: www.cs.ucr.edu/~eamonn/time_series_data/
11. Muhammad Fuad, M.M.: ABC-SG: A New Artificial Bee Colony Algorithm-Based Distance of Sequential Data Using Sigma Grams. The Tenth Australasian Data Mining Conference - AusDM 2012, Sydney, Australia, 5-7 December, (2012)
12. Muhammad Fuad, M.M.: Differential Evolution versus Genetic Algorithms: Towards Symbolic Aggregate Approximation of Non-normalized Time Series. Sixteenth International Database Engineering & Applications Symposium– IDEAS'12 , Prague, Czech Republic,8-10 August, 2012 . Published by BytePress/ACM (2012)
13. Muhammad Fuad, M.M.: Particle Swarm Optimization of Information-Content Weighting of Symbolic Aggregate Approximation. The 8th International Conference on Advanced Data Mining and Applications -ADMA2012, 15-18 December 2012, Nanjing, China. Published in LNCS/LNAI (2012)
14. Muhammad Fuad, M.M.: Towards Normalizing the Edit Distance Using a Genetic Algorithms–Based Scheme. In: Zhou, S., Zhang, S., Karypis, G. (eds.) ADMA 2012. LNCS, vol. 7713, pp. 477–487. Springer, Heidelberg (2012)
15. Muhammad Fuad, M.M.: Using Differential Evolution to Set Weights to Segments with Different Information Content in the Piecewise Aggregate Approximation. In: 16th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems, KES 2012, San Sebastian, Spain, September 10-12. Frontiers of Artificial Intelligence and Applications (FAIA). IOS Press (2012)
16. Zezula et al., :Similarity Search - The Metric Space Approach, Springer (2005).