A hybrid of bacterial foraging and differential evolution -based distance of sequences

Fuad, M. M. M.

Original citation:
Fuad, MMM 2014, 'A hybrid of bacterial foraging and differential evolution -based distance of sequences' Procedia Computer Science, vol. 35, pp. 101-110.
https://dx.doi.org/10.1016/j.procs.2014.08.089

DOI 10.1016/j.procs.2014.08.089
ISSN 1877-0509
ESSN 1877-0509

Publisher: Elsevier

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 35 (2014) 101 – 110

Procedia Computer Science

18th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems - KES2014

A hybrid of bacterial foraging and differential evolution -based distance of sequences

Muhammad Marwan Muhammad Fuad*

The University of Tromsø - The Arctic University of Norway, NO-9037 Tromsø, Norway

Abstract

In a previous work we presented a new distance that we called the sigma gram distance, which is used to compute the similarity between two sequences. This distance is based on parameters which we computed through an optimization process that used the artificial bee colony; a bio-inspired optimization algorithm. In this paper we show how a hybrid of two optimization algorithms; bacterial foraging and differential evolution, when used to compute the parameters of the sigma gram distance, can yield better results than those obtained by applying artificial bee colony. This superiority in performance is validated through experiments on the same data sets to which artificial bee colony, on the same optimization problem, was tested.

Peer-review under responsibility of KES International.

Kewwords: Bacterial Foraging, Differential Evolution, Sigma Gram Distance.

1. Introduction

Similarity search is an important problem in computer science. This problem has several applications in many domains such as text, video and image retrieval, pattern recognition, bioinformatics, web search, fingerprint databases, and many others. The similarity search problem involves a collection of data objects D and a given query q. The task of this problem is to retrieve the data objects in D which are "close" to q

* Corresponding author. Tel.: +47 77 64 64 73.
E-mail address: marwan.fuad@uit.no.

Peer-review under responsibility of KES International.
doi:10.1016/j.procs.2014.08.089

according to some semantics of closeness. This closeness or similarity between two data objects is measured using a *similarity measure* or when satisfying certain axioms it is usually called a *distance metric*. The metric model of handling the similarity search problem has been widely used with different data types.

Time series data is one of the data types to which the metric model is frequently (but not always) applied to handle different tasks such as classification and clustering.

Time series are high-dimensional data, so there are different techniques to reduce their dimensionality, among these techniques symbolic methods have attracted particular attention.

The main distance used to compare two strings is the *Edit Distance* (ED) [1] which is defined as the minimum number of delete, insert, and change operations needed to transform string S into string T. However, this distance has its limitations because it considers local similarity only.

In [2] we presented a new distance metric, the *Sigma Gram* distance (SG) that is applied to sequences. SG uses parameters which we computed using an optimization algorithm called *Artificial Bee Colony* (ABC).

Bio-inspired, also called *nature-inspired*, optimization is a rapidly growing domain of research, and new algorithms emerge constantly. Yet these algorithms may have their own shortcomings in certain optimization problems. One of the new techniques that have been successfully used to overcome these drawbacks of optimization algorithms is to couple two optimizers to produce a hybrid one that has the advantages of each of the optimizers.

In this paper we use a hybrid of two bio-inspired optimization algorithms to compute the parameters of SG and we show how the hybrid algorithm outperforms the aforementioned ABC algorithm.

The rest of this paper is organized as follows: Section 2 presents related work. The hybrid algorithm is introduced in Section 3, and the comparison with ABC is conducted in Section 4. We conclude this paper in Section 5.

2. Related work

Strings, also called *sequences* or *words*, are a way of representing data. This data type exists in many fields of computer science such as molecular biology where DNA sequences are represented using four nucleotides which correspond to the four bases: adenine (A), cytosine (C), guanine (G) and thymine (T). This can be expressed as a 4-symbol alphabet. Protein sequences can also be represented using a 20-symbol alphabet which corresponds to the 20 amino acids. The edit distance [1] is the main distance that is applied to compute the similarity between two strings. It is defined as the minimum number of delete, insert, and substitute operations needed to transform one sequence *S* into another sequence *T*.

In a previous work [2] we presented an extension of the edit distance, which is based on the sum of *n*-grams. The proposed distance SG is defined as follows:

Let Σ be a finite alphabet, and let Σ^* be the set of strings on Σ. Given *n*, Let $f_{a_n}^{(S)}$ be the frequency of the *n-gram* a_n in S, and $f_{a_n}^{(T)}$ be the frequency of the *n-gram* a_n in T, where S, T are two strings in Σ^*. Let $\mathbf{N}$ be the set of integers, and $\mathbf{N}^+$ the set of positive integers.

Let $g : \mathbf{N}^+ \times \Sigma^* \rightarrow \mathbf{N}$

$$g(n,S) = n \quad \text{if} \quad 1 \le n \le |S|$$
$$g(n,S) = |S| + 1 \quad \text{if} \quad |S| < n$$

Then SG is defined as:

$$SG(S,T) = \sum_{n=1}^{max(|S|,|T|)} \lambda_n \cdot \left[|S| + |T| - g(n,S) - g(n,T) + 2 = 2 \cdot \sum_{a_n \in \Sigma^n} min\left(f_{a_n}^{(S)}, f_{a_n}^{(T)}\right) \right] \quad (1)$$

where $|S|, |T|$ are the lengths of the two strings S, T respectively, and where $\lambda_n \in \mathbf{R}^+ \cup \{0\}$.

Determining the value of the parameters λ_n is not a trivial task. In [2] the value of λ_n was calculated as the output of an optimization problem. The optimization algorithm we used was artificial bee colony (ABC) [3], which is a bio-inspired optimization algorithm based on the foraging behavior of bees. In ABC each food source represents a potential solution of the optimization problem and the quality of the food represents the value of the objective function to be optimized. The control parameters of ABC are the population size (the number of food sources) *pop_size* , the number of cycles *nr_cycles* , and the number of trials of a certain food source *max_nr*.

3. A Synergy of Bacterial Foraging and Differential Evolution

Different optimization algorithms have different qualities. The principle of hybridization is to benefit from this fact by combining two optimization algorithms to obtain a new optimization algorithm that takes advantage of the strengths of the two methods. In this paper we are concerned with the hybridization of two optimization algorithms; *Differential Evolution* (DE) and *Bacterial Foraging* (BF).

3.1. Differential evolution

Differential Evolution (DE) is a bio-inspired optimization algorithm based on the principles of genetics and natural selection. DE is 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 a function which is usually called 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 *pop_size*. Each of these solutions is a vector of *p* dimensions and it represents a possible solution to the optimization problem. The fitness function of each individual is computed. In the next step for each individual of the population, which we call the *target vector* $\vec{T}_i$ at this phase, 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. 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 the interval $[0,1[$.

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. In the following we present the crossover scheme presented in [5] which we adopt in this paper; an integer *Rnd* is randomly chosen among the dimensions $[1, p]$. This guarantees that at least one of the dimensions will be changed. Then the trial vector $\vec{R}$ is formed as follows:

$$t_i \quad \begin{cases} t_{i,r1} + F\left(t_{i,r2} - t_{i,r3}\right) = & if \quad \left(\bar{r}and_{i,j}\left[0,1\right[< \bar{C}_r\right) \vee \left(Rnd \quad i\right) = \\ \bar{\bar{t}}_{i,j} & otherwise \end{cases} \tag{2}$$

where $i \quad 1,..., p$. 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.

3.2. Bacterial foraging

Bacterial Foraging (BF) is an optimization algorithm which is inspired by the foraging behavior of the *Escherichia coli* (*E. coli*) bacteria. The principle of BF is that natural selection tends to eliminate animals with poor foraging strategies and either replaces them with others that have better foraging strategies or shapes them into ones which have these desirable strategies [6]. BF formulates this process as an optimization problem.

In the following we present a brief description of BF taken mainly from [7]; during foraging locomotion of *E. coli* is achieved by a set of flagella which, when rotating in the clockwise direction, cause the bacterium to *tumble*, and when rotating in the counterclockwise direction enable the bacterium to *swim.* These two movements are known as *chemotaxis*. Figure 1 shows the tumbling and swimming chemotactic movements. The purpose of chemotaxis is to help the bacterium approach or avoid nutrient or noxious substance gradients. Sudden environmental changes may destroy the chemotactic progress causing the elimination and dispersal of a group of bacteria.

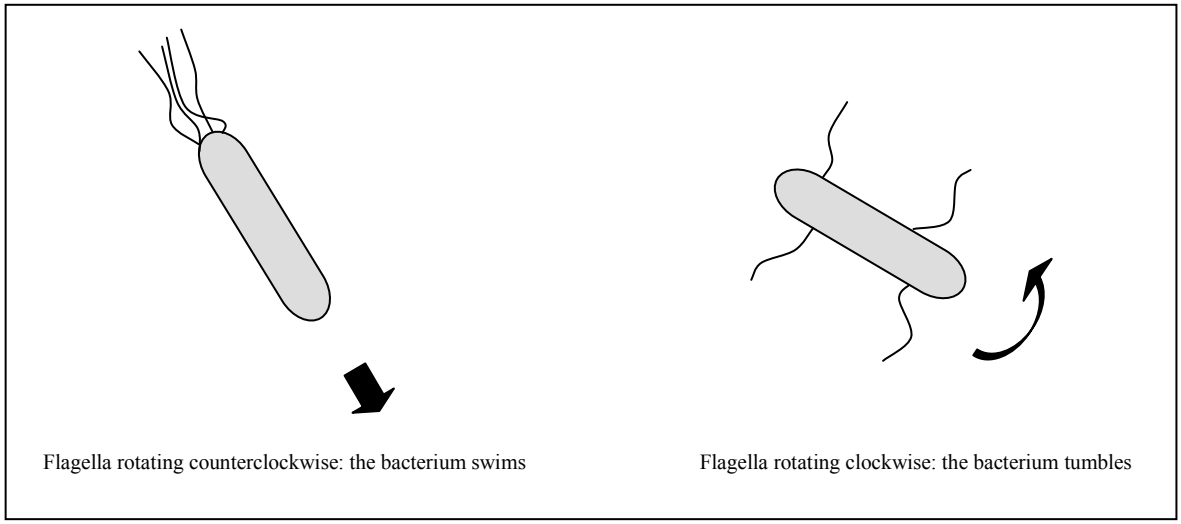

Fig. 1. Chemotactic movement: (a) swimming (b) tumbling

N_b	The number of bacteria in the population
N_c	The number chemotactic steps
N_s	The swimming length
N_{re}	The number of reproduction steps
N_{ed}	The number of elimination-dispersal events
P_{ed}	The probability of elimination-dispersal
$C(i)$	The size of the step taken in the random direction determined by the tumble

Table 1. The symbols used in the description of bacterial foraging

Given a function $f(\theta); \theta \in \mathbf{R}^p$ (p is the number of parameters) to be minimized. BF finds the minimum of f by applying four mechanisms; chemotaxis, swarming, reproduction, and elimination-dispersal, which we will illustrate shortly, but let us first present a few definitions which are necessary to understand these mechanisms: a *chemotactic step* is a tumble followed by another tumble, or a tumble followed by a swim. Table 1 summarizes the symbols we are going to use to describe BF.

The position of each member of the population of N_b bacteria at the j^{th} chemotactic step, k^{th} reproduction step, and l^{th} elimination-dispersal event is denoted by $P(i, j, k) = \{\theta^i(j,k,l) \mid i = 1,2,...,N_b\}$.

We now describe the four mechanisms we mentioned earlier in this section:

- **Chemotaxis**: Let $\theta^i(j,k,l)$ be the i^{th} bacterium at the j^{th} chemotactic step, k^{th} reproduction step, and l^{th} elimination-dispersal event, then the movement of the bacterium can be represented by :

$$\theta^i(j+1,k,l) = \theta^i(j,k,l) + C(i)\frac{\Delta(i)}{\sqrt{\Delta^T(i)\Delta(i)}} \tag{3}$$

where Δ is a vector in the random direction whose elements lie in the interval [*-1*, *1*].

- **Swarming**: *E. coli* demonstrate a swarming behavior in that they travel in rings of bacteria which move up the nutrient medium when they are placed in the center of a semisolid matrix with a single nutrient chemo-effecter. When simulated by a high level of succinate the bacteria release an attractant aspartate which helps them aggregate into groups and thus move as a swarm. The cell-to-cell signal in the swam can be represented by the following function:

$$f_{cc}(\theta, P(j,k,l)) = \sum_{i=1}^{N_b} f_{cc}(\theta, \theta^i(j,k,l)) = \sum_{i=1}^{N_b}\left[-d_{attractant}.exp\left(-\omega_{attractant}\sum_{m=1}^{p}(\theta_m - \theta_m^i)^2\right)\right] + \sum_{i=1}^{N_b}\left[-h_{repellant}.exp\left(-\omega_{repellant}\sum_{m=1}^{p}(\theta_m - \theta_m^i)^2\right)\right] \tag{4}$$

where $d_{attractant}$, $\omega_{attractant}$, $h_{repellant}$, $\omega_{repellant}$ are coefficients to be chosen by the algorithm designer.

The objective function $f_{cc}(\theta, P(j,k,l))$ is to be added to the original objective function to present a *time varying* objective function in that if many cells come close together there will be a high amount of attractant

and hence an increasing likelihood that other cells will move towards the group. This produces the swarming effect [6].

- = **Reproduction:** Through this process the least healthy bacteria die out and the healthier ones will undergo cell division to produce two daughter bacteria. This guarantees that the swam size will remain stable.

- **Elimination and dispersal:** There might be a gradual or sudden change in the environment where the bacteria live. As a result, the bacteria in a certain region are killed or a group might be dispersed into another location. This has two effects on chemotaxis; the first is destroying the chemotactic progress. The second is that the new bacteria might be placed at locations with a better food source, thus assisting chemotaxis.

3.3. Hybridization of BF and DE

Compared with other bio-inspired optimization algorithms, BF possesses a poor convergence behavior over multi-modal and rough fitness landscapes. Its performance is also heavily affected with the growth of problem dimensionality [8]. On the other hand, DE may suffer from *stagnation*; i.e. the inability of progressing towards global optima. DE may also suffer from premature convergence. To overcome these problems the authors of [9] proposed an optimization algorithm, called *Chemotactic Differential Evolution* (CDE), which is based on hybridizing DE and BF by integrating some features from both of these optimizers. The experiments conducted have shown that CDE outperforms both DE and BF.

In CDE each trial vector first undergoes an adaptive computational chemotaxis. The trial vector is viewed as an *E. coli* bacterium. During chemotaxis, the bacterium which is close to a noxious substance takes a larger chemotactic step to move towards nutrient substances. Before each move, it is ensured that the bacterium moves in the direction of increasing nutrient substance concentration; i.e. a region with smaller objective function value. After this, it is subjected to DE mutation. For the trial vector, three vectors, other than the previous one, are selected, one of which is added with a scaled difference of the remaining two. The produced vector probabilistically interchanges its components with the original vector. Offspring vector replaces the original one if the objective function value is smaller for it. The process is repeated several times over the entire population in order to obtain the optimal solution [9].

4. Performance evaluation

The aim of our experiments is to compare the performance of CDE with that of ABC which we presented in [2] on the same optimization problem which is a classification task of symbolically represented time series.

A *time series S* is an ordered collection:

$$S \quad \{(t_1,v_1),(t_2,v_2),...,(t_n,v_n)\} \tag{5}$$

where $t_1 < t_2 < ... < t_n$, and where v_i are the values of the observed phenomenon at time points t_i.

Time series data mining handles several tasks such as classification, clustering, similarity search, motif discovery, anomaly detection, and others. Time series are high-dimensional data so they are usually processed by using representation methods that are used to extract features from these data and project them on lower-dimensional spaces.

The *Symbolic Aggregate approXimation* method (SAX) [10] is one of the most important representation methods of time series. SAX is applied as follows:

1-The time series are normalized.

2-The dimensionality of the time series is reduced using PAA [11, 12].

3-The PAA representation of the time series is discretized by determining the number and location of the breakpoints. Their locations are determined using Gaussian lookup tables. The interval between two successive breakpoints is assigned to a symbol of the alphabet, and each segment of PAA that lies within that interval is discretized by that symbol.

The last step of SAX is using the following similarity measure:

$$MINDIST(\hat{S},\hat{R}) \equiv \sqrt{\frac{n}{N}} \sqrt{\sum_{i=1}^{N} (dist(\hat{s}_i, \hat{r}_i))^2} \quad (6)$$

Where n is the length of the original time series, N is the length of the strings (the number of the segments), $\hat{S}$ and $\hat{R}$ are the symbolic representations of the two time series S and R, respectively, and where the function $dist(\ )$ is implemented by using the appropriate lookup table.

The objective function of the optimization problem of our experiments is the error of a time series classification task based on the first nearest-neighbor (*1*-NN) rule using leaving-one-out cross validation. This means that every time series is compared to the other time series in the dataset. If the *1*-NN does not belong to the same class, the error counter is incremented by 1. The parameters of the optimization problem are λ_n in relation (1), in other words, we compute λ_n that minimize the classification error using ABC as an optimizer (which is the optimizer used in [2]) which we refer to as ABC-SG, and compare that with the optimal values of λ_n when those λ_n values are obtained by using CDE as an optimizer in (1), which we refer to as CDE-SG.

In our experiments we used the same datasets on which ABC-SG was tested in [2]. These datasets are available at UCR [13].

As indicated earlier, the tested methods used symbolically represented time series This means that the time series were transformed to symbolic sequences using the first three steps of SAX presented earlier in this section, but instead of using MINDIST given in relation (6), we use ABC-SG (or CDE-SG). The parameters λ_n in relation (1) are computed using ABC (or CDE). This means, for each value of the alphabet size we formulate an ABC (or CDE) optimization problem where the fitness function is the classification error, and the parameters of the optimization problem are λ_n. Practically n can take any value that does not exceed that of the shortest string of the two strings S, T. However, in the experiments we conducted $n \in \{1,2,3\}$ because these are the values of interest for time series.

The control parameters for CDE were the following; the number of bacteria in the population N_b is 20, the differentiation constant F was set to 0.9, and the crossover constant C_r was set to 0.5. The number of chemotactic steps N_c was set to 5, the swimming length N_s was set to 4. The number of reproduction steps N_{re} was 4, the number of reproduction steps N_{ed} was 2, and P_{ed}, the elimination-dispersal probability, was set to 0.25. The dimension of the problem p is that of n. As for λ_n, their values are in fact unconstrained, but for simplicity we optimized them in the interval $[0,2]$.

As for ABC, the control parameters were the same used in [2]; the number of cycles *nr_cycles* was set to 20, and the number of trials of a certain food source *max_nr* was set to 10. As for the population size, it is the same as that for CDE.

Beef

	ABC-SG			CDE-SG		
	n=1	n=2	n=3	n=1	n=2	n=3
α*=3	0.567	0.567	0.567	0.533	0.533	0.500
α =10	0.500	0.500	0.467	0.467	0.467	0.433
α =20	0.333	0.367	0.367	0.333	0.333	0.333

(*: α is the alphabet size)

Coffee

	ABC-SG			CDE-SG		
	n=1	n=2	n=3	n=1	n=2	n=3
α =3	0.571	0.393	0.357	0.393	0.357	0.357
α =10	0.214	0.286	0.214	0.214	0.179	0.179
α =20	0.143	0.071	0.143	0.143	0.071	0.071

ECG200

	ABC-SG			CDE-SG		
	n=1	n=2	n=3	n=1	n=2	n=3
α =3	0.190	0.210	0.240	0.180	0.210	0.220
α =10	0.200	0.220	0.220	0.200	0.210	0.210
α =20	0.230	0.230	0.260	0.220	0.220	0.250

Gun_Point

	ABC-SG			CDE-SG		
	n=1	n=2	n=3	n=1	n=2	n=3
α =3	0.193	0.193	0.180	0.180	0.180	0.146
α =10	0.146	0.127	0.133	0.133	0.127	0.120
α =20	0.087	0.073	0.073	0.053	0.053	0.067

FaceFour

	ABC-SG			CDE-SG		
	n=1	n=2	n=3	n=1	n=2	n=3
α =3	0.057	0.057	0.057	0.057	0.045	0.045
α =10	0.045	0.057	0.114	0.045	0.045	0.068
α =20	0.114	0.114	0.102	0.090	0.102	0.102

OSULeaf

	ABC-SG			CDE-SG		
	n=1	n=2	n=3	n=1	n=2	n=3
α =3	0.351	0.343	0.331	0.331	0.331	0.322
α =10	0.298	0.306	0.298	0.298	0.298	0.298
α =20	0.322	0.331	0.331	0.306	0.322	0.322

Table 2. Comparison between ABC-SG and CDE-SG

For each dataset we first the apply the optimizer (ABC or CDE) on the training datasets to get the vector λ_n that minimizes the classification error on these training datasets, then we utilize this optimal $\lambda_{\overline{n}}$-vector on the corresponding testing datasets to get the final classification error for each dataset.

In Table 2 we present some of the results we obtained for alphabet size equal to 3, 10, and 20, respectively, which were the values on which ABC-SG was tested.

The results show that the classification errors of CDE-SG, and for all the datasets shown, are equal or smaller than those of ABC-SG and for all values of the alphabet size.

5. Conclusion

In this paper we applied a hybrid optimization algorithm; chemotactic differential evolution - CDE, to compute the parameters $\lambda_{\overline{n}}$-of SG distance, which minimize the error of a time series classification task. We compared this optimizer with another one; artificial bee colony- ABC, and we showed experimentally that CDE gives better results.

This paper shows the advantages of coupling optimization methods to produce a new hybrid method to improve the performance of stand-alone optimizers. However, the resulting hybrid method may have too many control parameters that more research should be conducted to reduce the number of these control parameters.

References

1. Wagner, R.A., Fischer, M. J. : The string-to-string correction problem, *Journal of the Association for Computing Machinery*, Vol. 21, No. I, January 1974, pp. 168-173.
2. Muhammad Fuad, M.M. (2012): 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.
3. Karaboga, D. (2005): An idea based on honey bee swarm for numerical optimization. *Technical Report TR06*, Erciyes University, Engineering Faculty, Computer Engineering Department.
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. Feoktistov, V. (2006): *Differential evolution: in search of solutions* (Springer Optimization and Its Applications). Secaucus, NJ, USA: Springer- Verlag New York, Inc.
6. Passino, K. M.: Biomimicry of bacterial foraging for distributed optimization and control, *IEEE Control Syst. Mag.*, vol. 22, no. 3, Jun. 2002, pp. 52–67.
7. Das, S., Biswas, A., Dasgupta, S., Abraham, A.: Bacterial foraging optimization algorithm: theoretical foundations, analysis, and applications, *foundations of computational intelligence. Global optimization, studies in computational intelligence*, vol 3, 2009, pp 23–55.
8. Biswas, A., Dasgupta, S., Das, S. and Abraham, A. (2007) : Synergy of PSO and bacterial foraging optimization: a comparative study on numerical benchmarks. *Second International Symposium on Hybrid Artificial Intelligent Systems* (HAIS 2007), Advances in Soft

computing Series, Springer Verlag, Germany, Corchado, E. et al. (Eds.): Innovations in Hybrid Intelligent Systems, ASC 44, pp. 255-263.

9. Biswas, A., Dasgupta, S., Das, S., and Abraham, A.: A Synergy of differential evolution and bacterial foraging algorithm for global optimization. *Neural New World*, vol. 17, no. 6, 2007, pp. 607–626.
10. Lin, J., Keogh, E., Lonardi, S., Chiu, B. Y. (2003) :A symbolic representation of time series, with implications for streaming algorithm. *DMKD 2003*: 2-11.
11. Keogh, E,. Chakrabarti, K,. Pazzani, M. & Mehrotra: Dimensionality reduction for fast similarity search in large time series databases. *J. of Know. and Inform. Sys.* 2000.
12. Yi, B,K., & Faloutsos, C. (2000): Fast time sequence indexing for arbitrary Lp norms. *Proceedings of the 26st International Conference on Very Large Databases*, Cairo, Egypt .
13. Keogh, E., Zhu, Q., Hu, B., Hao. Y., Xi, X., Wei, L. & Ratanamahatana, The UCR Time Series Classification/Clustering Homepage: www.cs.ucr.edu/~eamonn/time_series_data/ C. A. 2011.