A Synergy of Artificial Bee Colony and Genetic Algorithms to Determine the Parameters of the Σ-gram Distance

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: In a previous work we presented the Σ-gram distance that computes the similarity between two sequences. This distance includes parameters that we calculated by means of an optimization process using artificial bee colony. In another work we showed how population-based bio-inspired algorithms can be sped up by applying a method that utilizes a pre-initialization stage to yield an optimal initial population. In this paper we use this pre-initialization method on the artificial bee colony algorithm to calculate the parameters of the Σ-gram distance. We show through experiments how this pre-initialization method can substantially speed up the optimization process.

Keywords: Artificial Bee Colony, Bio-inspired Optimization, Genetic Algorithms, Pre-initialization, Σ-gram.

1 Introduction

Optimization is a rich domain of research and application in computer science and applied mathematics. An optimization problem can be defined as follows: Given a function $f : U \subseteq \mathrm{R}^{nbp} \rightarrow \mathrm{R}$ (*nbp* is the number of parameters), find the solution $\overrightarrow{X^{*}} = \left[x_{1}^{*}, x_{2}^{*}, \ldots, x_{nbp}^{*}\right]$ 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*, or the *objective function*. Informally, the purpose of an optimization process is to find the best-suited solution of a problem subject to given constraints.

Bio-inspired, also called *nature-inspired*, optimization algorithms have gained popularity in many applications because they handle a variety of optimization problems. These algorithms are inspired by natural processes, natural phenomena, or by the collective intelligence of natural agents.

One of the main bio-inspired optimization families is *Evolutionary Algorithms* (*EA*). *EA* are population-based metaheuristics that use the mechanisms of Darwinian evolution. The *Genetic Algorithm* (*GA*) is the main member of *EA*. *GA* is an optimization and search technique based on the principles of genetics and natural selection [1]. GA has the following elements: a population of individuals, selection according to fitness, crossover to produce new offspring, and random mutation of new offspring [5].

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 [8] [9] [10].

In [6] 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*); one of the bio-inspired optimization algorithms.

Applying *ABC*, and other bio-inspired algorithms, to the data mining problems requires recruiting extensive computing resources and long computational time. This is a part of what is called *expensive optimization*. In [7] we presented a new technique to handle such optimization problems. In this work we re-visit the work presented in [6] and apply the technique we introduced in [7] to speed up the optimization process. The rest of this paper is organized as follows: Section 2 is a background section, in Section 3 we explain the pre-initialization method and we show how it can be used to speed up the optimization process, we test this pre-initialization method to compute the parameters of the Σ-gram in Section 4. Section 5 is a concluding section.

2 Background

Let Σ be a finite alphabet of a set of characters. A *string* is an ordered set of this alphabet. Strings appear in a variety of domains in computer science and bioinformatics. The *Edit Distance* is the main distance used to compare two strings. In a previous work [6] we presented an extension of the edit distance, which is based on the sum of *n*-grams. The proposed distance Σ-gram (which we refer to in this paper as *SG*) is defined as follows:

Let Σ^* be the set of strings on Σ. Given a positive integer *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 $\mathbb{N}$ be the set of integers, and $\mathbb{N}^+$ the set of positive integers.

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

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

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

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 $\lambda_n \in \mathbb{R}^+ \cup \{0\}$.

Determining the values of the parameters λ_n is not a trivial task. In [6] these values were obtained as the outcome of an optimization problem. The optimization algorithm we used was artificial bee colony (*ABC*).

3.1 Artificial Bee Colony (*ABC*)

Artificial Bee Colony (*ABC*) [2] is an optimization algorithm inspired by the foraging behavior of bees. In *ABC* each food source represents a potential solution to the optimization problem and the quality of the food represents the value of the objective function to be optimized. Artificial bees explore and exploit the search space. These bees communicate and share information about the location and quality of food sources. Bees exchange of information by performing a *waggle dance* which takes place in the dancing area in the hive. In *ABC* there are three kinds of bees; *employed bees*: these are the bees that search in the neighborhood of a food source. They perform a dance with a probability that is proportional to the quality of the food source, *onlooker bees*: these bees are found on the dance floor, and *scouts*: these bees explore the search space randomly.

The first step of *ABC* is generating a randomly distributed population of size (*pop_size*) of food sources which correspond to potential solutions. Each solution $\vec{x}_i, i \in \{1,.., pop\_size\}$ is a vector whose dimension is (*nr_par*) which is equal to the number of parameters of the function f to be optimized. The population is subject to change for a number of cycles (*nr_cycles*). In each cycle every employed bee perturbs the current solution using a local search procedure. The perturbation produces a new solution:

$$\vec{x}_i^* = \vec{x}_i + rand(-1,1)(\vec{x}_i - \vec{x}_k) \quad ,i \neq k \qquad (2)$$

The above relation is not applied to all parameters but only to a certain number of them. The parameters to be altered are chosen randomly. The algorithm uses a greedy selection to decide if the new solution should be kept or discarded, i.e.:

$$\vec{x}_i = \begin{cases} \vec{x}_i^* & if \quad f(\vec{x}_i^*) < f(\vec{x}_i) \\ \vec{x}_i & otherwise \end{cases} \qquad (3)$$

After all employed bees have modified their positions the onlooker bees choose one of the current solutions depending on a probability that corresponds to the fitness value of that solution according to the following rule:

$$p_i = \frac{f(\vec{x}_i)}{\sum_{k=1}^{pop\_size} f(\vec{x}_k)} \qquad (4)$$

After that the onlooker bees try to improve the solution using the same mechanism that was described in (4). The number of trials the algorithm attempts to improves the same solution is limited by a maximum number (*max_nr*) after which the solution is abandoned and the bees employed by that food source become scouts. The abandoned solution is replaced by a new solution found by the scouts.

3 A Pre-initialized Artificial Bee Colony Algorithm

The optimization problem we presented in Section 2 requires extensive computing. This type of optimization problems is called expensive optimization. In [7] we introduced a new method that can be applied to any population-based optimization algorithm to speed up the optimization process. The principle of this method is to use an "optimal" initial population by adding to the main problem, which we call *MainOptim*, an artificial optimization problem to optimize the initial population. We call this latter problem *SecOptim*. As a fitness function of *SecOptim* we choose one that gives as much information as possible about the search space of *MainOptim* since this initial population will eventually be used to optimize *MainOptim*. The fitness function for *SecOptim* will be the one that maximizes the average distance of the chromosomes of the population, i.e.:

$$f_{secOptim} = \frac{2}{secPopSize(secPopSize-1)} \sum_{i=1}^{secPopSize-1} \sum_{j=i+1}^{secPopSize} d(ch_i, ch_j) \qquad (5)$$

where *secPopSize* is the population size of *SecOptim* , *ch* is the chromosome. *d* is a distance, which we choose to be the Euclidean distance. Notice that $d(ch_i, ch_j) = d(ch_j, ch_i)$ so we only need to take half of the summation in (5).

The other component of *SecOptim* is the search space. As indicated earlier, *SecOptim* is a separate optimization problem from *MainOptim* with its own search space. The search space of *SecOptim* is a discrete one whose points are feasible solutions of *MainOptim*. In other words, the search space of *SecOptim* is a *pool* of solutions of *MainOptim* . The cardinality of this pool is denoted by *poolSize*.

Now all the elements of *SecOptim* are defined. *poolSize* is a new element that is particular to our method. In the experimental section we discuss this element further.

As we can see, *MainOptim* and *SecOptim* are two independent problems, so we can use two different optimization algorithms. One of the optimization algorithms that we can use for *SecOptim* is the Genetic Algorithms for its exploiting ability.

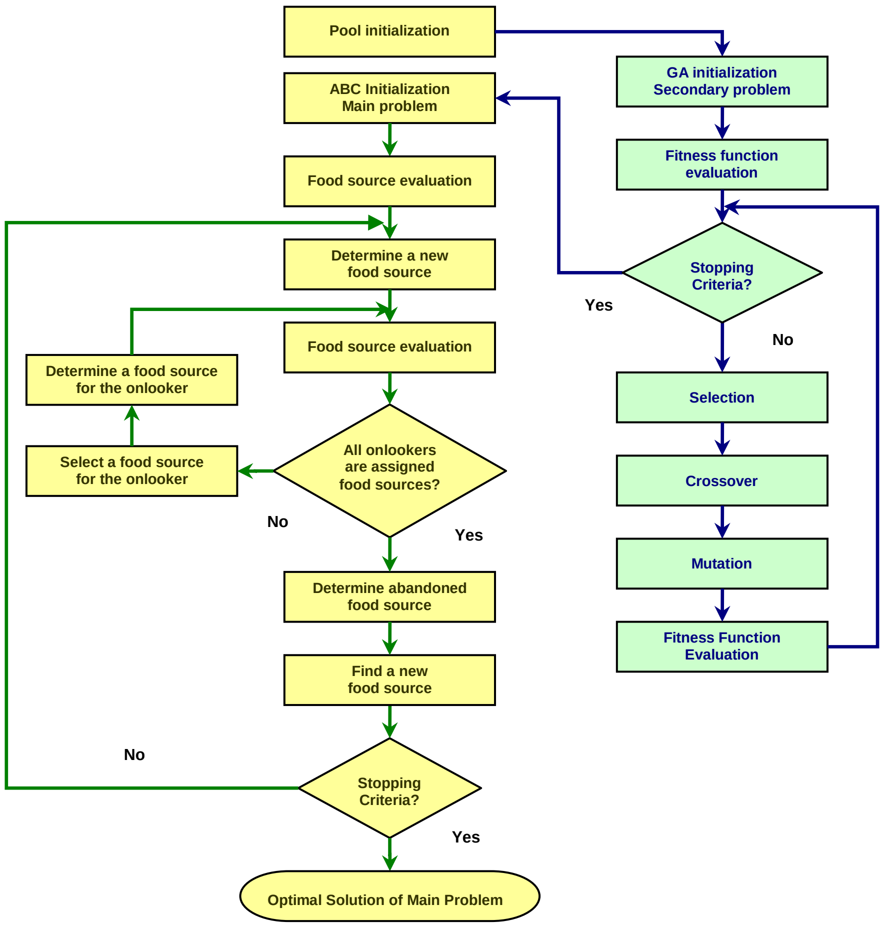
Pool initialization
ABC Initialization Main problem
Food source evaluation
Determine a new food source
Food source evaluation
Determine a food source for the onlooker
Select a food source for the onlooker
All onlookers are assigned food sources?
No
Yes
Determine abandoned food source
Find a new food source
No
Stopping Criteria?
Yes
Optimal Solution of Main Problem
GA initialization Secondary problem
Fitness function evaluation
Stopping Criteria?
Yes
No
Selection
Crossover
Mutation
Fitness Function Evaluation

Fig. 1. Pre-initialized *ABC* algorithm

The Genetic Algorithm (*GA*): *GA* is a widely-known bio-inspired optimization algorithm. *GA* starts by randomly generating a number of chromosomes. This step is called *initialization*. The fitness function of each chromosome is evaluated. The next step is *selection*. The purpose of this procedure is to determine which chromosomes are fit enough to survive. *Crossover* is the next step in which offspring of two parents are produced to enrich the population with fitter chromosomes. The last element is *Mutation* of a certain percentage of chromosomes.

The basis of our work we present here is that instead of applying *ABC* directly, which was the case in [6], to obtain the optimal values of λ_n , we use an optimal initial population by applying the method we presented in [7], and which we described in this section. Fig. 1 illustrates this enhanced optimization algorithm.

4 Empirical Evaluation

As in [6], we test the modified algorithm, which we refer to hereinafter as *ABC_SG*, with the new method presented in Section 3, which we refer to as *PreInitial_ABC_SG*, on a time series classification task. Time series are high-dimensional numeric data, but there are some techniques that reduce their dimensionality and transform them into series of characters. The *Symbolic Aggregate approXimation* method (*SAX*) [4] is the most important symbolic representation method of time series. *SAX* uses the following similarity measure:

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

Where n is the length of the original time series, N is the length of the strings (the number of 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.

As in [6] the objective function of our optimization problem is the classification error based on the first nearest-neighbor (*1*-NN) rule using leaving-one-out cross validation. 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_SG* as an optimizer, and compare that with the optimal values of λ_n when those λ_n values are obtained by using *PreInitial_ABC_SG* as an optimizer in (1)

In our experiments we used the same datasets on which *ABC_SG* was tested in [6]. These datasets are available at UCR [3]. The time series were represented symbolically using *SAX*, and then instead of applying (6) we apply *ABC_SG* (or *PreInitial_ABC_SG*).

We used 3 different values of the alphabet size in *SAX* : 3,10, 20. As for *n* in (1) we used $n \in \{1,2,3\}$. When testing a method, we first apply it to the training sets to obtain the optimal λ_n then these values are used on the testing sets.

The aim of the experiments is to show that by using *PreInitial_ABC_SG* , we can get classification errors close to those obtained by *ABC_SG* in a shorter time. This is achieved practically by running *PreInitial_ABC_SG* for a smaller number of generations; *NrGen*=20 (*MainOptim*), and running *ABC_SG* for *NrGen*=100. We then compare the results in term of classification error and

In Table 1 we show the classification error of *PreInitial_ABC_SG* and *ABC_SG* (because of space limitation we only show a part of the tested datasets).

Table 1. Comparison between the classification error of *PreInitial_ABC_SG* and that of *ABC_SG* for different values of the alphabet size and for different *n*-grams.

Dataset	**Method**		***n*-gram**		
			n=1	*n*=2	*n*=3
ECG	*PreInitial_ABC_SG*	α= 3	0.210	0.210	0.220
		α=10	0.200	0.210	0.220
		α=20	0.220	0.240	0.250
	ABS_SG	α= 3	0.190	0.210	0.240
		α=10	0.200	0.220	0.220
		α=20	0.230	0.230	0.260
Gun_Point	*PreInitial_ABC_SG*	α= 3	0.180	0.193	0.180
		α=10	0.133	0.133	0.127
		α=20	0.073	0.073	0.067
	ABS_SG	α= 3	0.193	0.193	0.180
		α=10	0.146	0.127	0.133
		α=20	0.087	0.073	0.073
FaceFour	*PreInitial_ABC_SG*	α= 3	0.057	0.057	0.045
		α=10	0.045	0.045	0.102
		α=20	0.090	0.114	0.102
	ABS_SG	α= 3	0.057	0.057	0.057
		α=10	0.045	0.057	0.114
		α=20	0.114	0.114	0.102
OSULeaf	*PreInitial_ABC_SG*	α= 3	0.331	0.343	0.322
		α=10	0.298	0.298	0.298
		α=20	0.306	0.343	0.322
	ABS_SG	α= 3	0.351	0.343	0.331
		α=10	0.298	0.306	0.298
		α=20	0.322	0.331	0.331

In Table 2 we present the wall clock time comparison between *PreInitial_ABC_SG* and *ABC_SG* for the datasets presented in Table 1. The experiments were conducted on Intel Core 2 Duo CPU with 3G memory.

Table 2. Run time comparison between *PreInitial_ABC_SG* and *ABC_SG*

Dataset	**Method**		***n*-gram**		
			n=1	*n*=2	*n*=3
ECG	*PreInitial_ABC_SG*	α= 3	01h 38m 43s	02h 00m 51s	03h 46m 25s
		α=10	01h 42m 08s	02h 08m 03s	03h 59m 35s
		α=20	01h 47m 26s	02h 53m 52s	34h 37m 22s
	ABS_SG	α= 3	08h 16m 49s	10h 06m 15s	18h 54m 17s
		α=10	08h 30m 24s	10h 30m 42s	19h 53m 23s
		α=20	08h 49m 34s	14h 23m 12s	154h 07m 20s
Gun_Point	*PreInitial_ABC_SG*	α= 3	01h 54m 06s	02h 22m 51s	04h 15m 34s
		α=10	02h 12m 17s	02h 38m 54s	04h 48m 22s
		α=20	02h 56m 48s	03h 34m 16s	42h 52m 32s
	ABS_SG	α= 3	09h 58m 34s	13h 16m 35s	21h 37m 18s
		α=10	11h 12m 41s	14h 52m 42s	24h 27m 52s
		α=20	14h 21m 26s	16h 46m 47s	138h 36m 62s
FaceFour	*PreInitial_ABC_SG*	α= 3	01h 22m 31s	01h 55m 24s	03h 25m 46s
		α=10	01h 38m 08s	02h 01m 45s	03h 14m 51
		α=20	01h 42m 52s	02h 43m 18s	29h 26m 26s
	ABS_SG	α= 3	07h 26m 52s	09h 57m 32s	17h 17m 52s
		α=10	08h 04m 28s	10h 03m 52s	19h 54m 26s
		α=20	08h 36m 26s	13h 52m 52s	148h 17m 53s
OSULeaf	*PreInitial_ABC_SG*	α= 3	12h 14m 52s	15h 45m 31s	25h 24m 35s
		α=10	13h 46m 26s	16h 52m 03s	23h 35m 43s
		α=20	15h 25m 26s	22h 12m 04s	184h 26m 54s
	ABS_SG	α= 3	62h 51m 02s	72h 26m 01s	81h 19m 31s
		α=10	64h 56m 41s	74h 55m 10s	88h 01m 47s
		α=20	67h 42m 45s	81h 57m 29s	543h 47m 29s

As we can see from Tables 1 and 2, *PreInitial_ABC_SG* is on average almost 5 times faster than *ABC_SG* although they both give quite comparable classification errors.

5 Conclusion

In this paper we applied a method from a previous work that speeds up population-based bio-inspired algorithms by initializing the optimization process using an optimal population. We applied this method to compute the parameters λ_n of the Σ-gram distance and we showed experimentally how by using an optimal initial population the optimization process can be sped up substantially.

References

1. Haupt, R.L., Haupt, S. E.: Practical Genetic Algorithms with CD-ROM. Wiley-Interscience (2004)
2. Karaboga, D.,: An idea based on honey bee swarm for numerical optimization. Technical Report TR06, Erciyes University, Engineering Faculty, Computer Engineering Department (2005)
3. 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)
4. Lin, J., Keogh, E., Lonardi, S., Chiu, B. Y.: A Symbolic Representation of Time Series, with Implications for Streaming Algorithms. DMKD 2003: 2-11(2003)
5. Mitchell, M.: An Introduction to Genetic Algorithms, MIT Press, Cambridge, MA (1996)
6. 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)
7. Muhammad Fuad, M.M.: A Pre-initialization Stage of Population-based Bio-inspired Metaheuristics for Handling Expensive Optimization Problems. The 9th International Conference on Advanced Data Mining and Applications -ADMA2013, December 14-16, 2013, Zhejiang, China. Published in LNCS/LNAI (2012)
8. 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)
9. Muhammad Fuad, M.M.: Towards Normalizing the Edit Distance Using a Genetic Algorithms–Based Scheme. The 8th International Conference on Advanced Data Mining and Applications -ADMA2012, 15-18 December 2012, Nanjing, China. Published in LNCS/LNAI (2012)
10. Muhammad Fuad, M.M.: Using Differential Evolution to Set Weights to Segments with Different Information Content in the Piecewise Aggregate Approximation. 16th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems – KES 2012– San Sebastian, Spain, September 10 - 12,2012. Published by IOS Press in "Frontiers of Artificial Intelligence and Applications (FAIA)" series (2012)