A Pre-initialization Stage of Population-based Bio-inspired Metaheuristics for Handling Expensive Optimization Problems

Muhammad Marwan Muhammad Fuad

Forskningsparken 3, Institutt for kjemi, NorStruct
The University of Tromsø - The Arctic University of Norway
NO-9037 Tromsø, Norway
mfu008@post.uit.no

Abstract: Metaheuristics are probabilistic optimization algorithms which are applicable to a wide range of optimization problems. Bio-inspired, also called nature-inspired, optimization algorithms are the most widely-known metaheuristics. The general scheme of bio-inspired algorithms consists in an initial stage of randomly generated solutions which evolve through search operations, for several generations, towards an optimal value of the fitness function of the optimization problem at hand. Such a scenario requires repeated evaluation of the fitness function. While in some applications each evaluation will not take more than a fraction of a second, in others, mainly those encountered in data mining, each evaluation may take up several minutes, hours, or even more. This category of optimization problems is called expensive optimization. Such cases require a certain modification of the above scheme. In this paper we present a new method for handling expensive optimization problems. This method can be applied with different population-based bio-inspired optimization algorithms. Although the proposed method is independent of the application to which it is applied, we experiment it on a data mining task.

Keywords: Bio-inspired Optimization, Differential Evolution, Expensive Optimization, Genetic Algorithms, Metaheuristics, Optimization Applications in Data Mining.

1 Introduction

Optimization is an important problem that has many applications. In an optimization problem we try to find a solution that minimizes or maximizes the value of a function that we call the *fitness function* or the *objective function*. Optimization problems can be discrete/ continuous/hybrid, constrained/unconstrained, single objective/ multiobjective, unimodal /multimodal. Optimization algorithms can be classified in several ways one of which is whether they are *single solution –based* algorithms or *population-based* algorithms. The term *metaheuristics* in the optimization literature refers to probabilistic optimization algorithms which are applicable to a large variety

of optimization problems. Many of these metaheuristics are inspired by natural processes hence the term *bio-inspired* or *nature-inspired* optimization algorithms. The general scheme used in all these algorithms is the following; an initial stage where a population of feasible solutions is randomly generated. The fitness function of these solutions is evaluated. The solutions with the highest values of the fitness function are favored and are given a higher possibility to survive the optimization process. The algorithm repeats for a certain number of *generations* or *cycles*, or it is terminated by a predefined *stopping criterion*.

As we can see from the above scheme, fitness function evaluation is a central part of bio-inspired optimization. While in some applications each evaluation will not take more than a fraction of a second, in others each evaluation may take up to several minutes, hours, or even more. This category of optimization is called *expensive optimization*. Such cases require a certain modification of the above scheme.

Data mining is a branch of computer science that handles several tasks, most of which demand extensive computing. As with other fields of research, different papers have proposed applying bio-inspired optimization to process data mining tasks [2], [3], [4], [5], [6], [7]. However, most of these applications are expensive optimization problems that require certain considerations.

In this paper we present a new method for handling expensive optimization problems. This method can be applied to different population-based bio-inspired optimization algorithms. Although the proposed method is independent of the application to which it is applied, we test it on a data mining task of setting weights for different segments of time series data according to their information content.

This paper is organized as follows: Section 2 is a background section. In Section 3 we present the new method. The experiments we conducted are reported in Section 4, and we conclude with Section 5.

2 Background

Although bio-inspired algorithms use different search strategies, they all share a common frame that is a based on the following steps:

. Initialization: In this step a collection of individuals (called *chromosomes*, *particles*, or *agents*, according to the algorithm) that represent a feasible solution to an optimization problem is generated randomly.

. Fitness Function Evaluation: The objective of this step is to rank the different, so-far examined, solutions of the problem, to determine their quality.

. Update: The term “update” here does not refer to the narrow meaning of it as it used in *Particle Swarm Optimization* (PSO), but it refers to a meta operation that directs the metaheuristics at iteration $t+1$ towards the region in the search space where a better solution is likely to be found. This update is based on the fitness evaluation at iteration t . This step is the abstract form of the *selection* step used in the *Evolutionary Algorithms* (EA) family.

. Mutation: This is a random alteration of a certain percentage of chromosomes. The objective of this operation is to allow the optimization algorithm to explore new regions in the search space.

. Iteration: This is not a step by itself, it is the repetition of the last three steps for a predefined number of times (*generations*, *cycles*, *iterations*, depending on the algorithm) which is usually predefined, or until the algorithm is terminated by a stopping criterion.

The performance of bio-inspired optimization highly depends on running the algorithm for a number of iterations sufficient to allow the solutions to evolve, otherwise the algorithm will, at best, only reach a local extreme point. The number of iterations, in turn, is dependent of the computational cost of fitness function evaluation. In general the number of fitness function evaluations can (roughly) be given by:

$$nEval = nItr.sPop \quad (\mathbf{1})$$

Where *nEval* is the number of fitness function evaluations, *nItr* is the number of iterations (generations) and *sPop* is the population size. We say that relation (1) is an approximate one because there are quite a number of variations; for instance, most algorithms will add to that relation another term related to the evaluations resulting from mutation, others will recycle evaluations from previous generations, etc.

One of the trivial techniques to handle expensive optimization problems is simply to reduce the number of generations *nItr*. While this may be acceptable to a certain degree, it could have serious consequences when *nItr* is drastically decreased. Bio-inspired optimization algorithms are supposed to mimic natural phenomena. For instance; EA simulate evolution of species as it happens over thousands of generations. This is the reason why many applications set *nItr* to 1000 or 2000 or even more. But when in some applications of expensive optimization *nItr* is set to 10, for instance, this changes the whole nature of the bio-inspired algorithm. At best, the algorithm in this case can only find a local extreme point, but in other cases the whole optimization process becomes meaningless. Besides, it is important to remember that the random initialization of the population assumes that the algorithm will be run for a certain number of generations enough to “erase” the effect of initialization of the population with specific chromosomes.

3 The Proposed Method

3.1 The Principle

One of the techniques that have already been proposed in bio-inspired optimization to avoid stagnating in a local extreme point is to run the algorithm several times, with different initial populations, and the best result of all these runs is kept. Although this approach is completely inappropriate for expensive optimization problems because it requires too many fitness function evaluations, our method stems from a similar idea; instead of running the algorithm several times, which is not computationally feasible, and instead of running the algorithm once for a limited number of iterations for expensive optimization problems, as has previously been discussed in Section 2, we

propose a new method that runs the algorithm for a limited number of iterations, but using an optimally initialized population.

3.2 Optimization of the Initial Population

As mentioned earlier, our method is based on running an expensive optimization algorithm for a small number of iterations but using an optimally-chosen initial population. However, we should keep in mind that this "optimality" of the initial population should not be determined by any evaluation of the expensive fitness function, otherwise the method would not make sense. The direct result of this requirement is that optimization of the initial population will be problem-independent. To put it simply; we have two separate and independent optimization problems; one is a sub-optimization problem, which is the problem of optimizing the initial population, we call this problem the *secondary optimization problem* and refer to it with (*SecOptim*), and the other is the original optimization problem with the expensive fitness function. We call this problem the *main optimization problem* and we refer to it with (*MainOptim*). *MainOptim* starts the optimization process with an optimal initial population obtained through *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*. This choice of our fitness function for *SecOptim* originates from one of the rules on which PSO is based, which is the rule of *separation* [8]. According to this rule each particle should avoid getting too close to its neighbors. The intuition behind this rule is that when two particles are close it is very likely that the value of the fitness function for both of them will not be very different. Based on the same intuition, between two different populations we have to choose the one whose chromosomes are as scattered as possible because such a population will give a better representation of the search space. Thus our choice for the fitness function for *SecOptim* will be the one that maximizes the average distance of the chromosomes of the population, i.e.:

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

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 $d\left(ch_i, ch_j\right) = d\left(ch_j, ch_i\right)$ so we only need to take half of the summation in (2).

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.

3.3 The Algorithm

Briefly, our method as described in Section 3.1 and 3.2, adds to the original optimization problem *MainOptim* another optimization problem *SecOptim* the outcome of which is the initial population of *MainOptim* . The aim of this process is to reduce the number of fitness function evaluations of *MainOptim* by starting the optimization process with an optimal initial population.

4 Application - Experiments

In this section we show how our algorithm is applied through an example of an optimization problem with an expensive fitness function. First we will present the problem and then we will discuss how our method is applied to it, and in the final part of this section we will conduct experiments to test our method.

4.1 The Problem

A *time series* is a collection of observations at intervals of time points. One key to mining time series data is to reduce their dimensionality so that they can be handled efficiently and effectively. Most time series data mining tasks require calculating the similarity between the time series. This similarity is quantified using a similarity measure or a distance metric. In [7] we presented a new distance of time series data, *WPAAD*, which is defined as:

$$WPAAD(S,R) = \sqrt{\frac{n}{N}}\sqrt{\sum_{i=1}^{N} w_i\left(\overline{s_i} - \overline{r_i}\right)^2}; \qquad w_i \in [0,1] \qquad (3)$$

Where n is the length of the time series, N is the number of frames in the reduced space, and where the time series are segmented into equal-sized frames, $\overline{s_i}$ ($\overline{r_i}$) is the mean of the data points S (R) that lie within that frame. The weights in (3) are set using the differential evolution (DE) which we present later in this paper. We called the dimensionality reduction technique based on this distance the *Differential Evolutionary Weighted Piecewise Aggregate Approximation* (*DEWPAA*)

4.2 Our Proposed Algorithm

The problem we presented in Section 4.1 is an example of expensive optimization problems, so we will use our algorithm, which we call the *PreInitialAlgo* to show that by starting the optimization process of *DEWPAA* with an optimized population resulting from our method, we can get the same results of *DEWPAA* but by a much smaller number of generations, thus with much fewer fitness function evaluations. In the language of our *PreInitialAlgo*, the optimization process of *DEWPAA* is *MainOptim*, and *SecOptim* is the optimization process that yields an optimal initial

population for *DEWPAA*. Since *MainOptim* and *SecOptim* are independent, we can use two different optimization algorithms if we wish to, so for our experiments we apply the Genetic Algorithms for *SecOptim* and the differential evolution for *MainOptim*. Figure 1 illustrates how *PreInitialAlgo* is applied. But let us first give a brief description of the Genetic Algorithms and the Differential Evolution.

The Genetic Algorithms (GAs): GAs are widely-known bio-inspired optimization algorithms. 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 Differential Evolution (DE): In DE for each individual, which we call the *target vector* $\vec{T}_i$, of the population we randomly choose three mutually distinct individuals; $\vec{V}_{r1}, \vec{V}_{r2}, \vec{V}_{r3}$ which combine to form the *donor vector* $\vec{D} = \vec{V}_{r1} + F\left(\vec{V}_{r2} - \vec{V}_{r3}\right)$. F is called the *mutation factor* . Then a *trial vector* $\vec{R}$ is formed from elements of $\vec{T}_i$ and $\vec{D}$. This includes utilizing another control parameter C_r called the *crossover constant*. In the next step $\vec{R}$ is compared with $\vec{T}_i$ to decide which one of them will survive in the next generation.

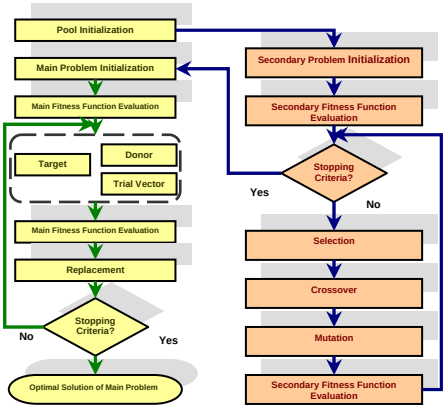

Fig. 1. A scheme of *PreInitialAlgo* using GA and DE

4.3 Experiments

We conducted our experiments on the datasets available at [1] to compare *PreInitialAlgo* with *DEWPAA*. For each tested data set we ran *PreInitialAlgo* for 20 generations to get the weights w_i in relation (3) that minimize the classification error of the training datasets, and then we used these optimal values w_i to classify the corresponding testing datasets to get the classification error. We repeated this process for three compression ratios 1:8, 1:12, and 1:16. We then ran *DEWPAA* for 100 generations to get w_i, also for the same compression ratios. The experiments were conducted on Intel Core 2 Duo CPU with 3G memory. We present in Table 1 the results of our experiments. As we can see in Table 1 the classification error of *PreInitialAlgo* is equal to, or even better than, that of *DEWPAA* even though the former is run only for 20 generations while the latter is run for 100 generations, which

Table 1. Comparison of classification accuracy between *PreInitialAlgo* and *DEWPAA* on different datasets for compression ratios 1:8, 1:12, and 1:16

Dataset	Method	Compression Ratios		
		1:8	1:12	1:16
Lighting7	PreInitialAlgo	0.397	0.260	0.479
	DEWPAA	0.438	0.384	0.479
MedicalImages	PreInitialAlgo	0.378	0.337	0.387
	DEWPAA	0.379	0.353	0.378
Lighting2	PreInitialAlgo	0.213	0.180	0.131
	DEWPAA	0.213	0.197	0.197
MALLAT	PreInitialAlgo	0.095	0.077	0.082
	DEWPAA	0.094	0.094	0.094
FaceUCR	PreInitialAlgo	0.240	0.302	0.364
	DEWPAA	0.238	0.316	0.366
FISH	PreInitialAlgo	0.194	0.246	0.200
	DEWPAA	0.194	0.240	0.229
synthetic_control	PreInitialAlgo	0.063	0.110	0.147
	DEWPAA	0.053	0.113	0.160

means that *DEWPAA* requires 5 times more fitness function evaluations than *PreInitialAlgo*, yet its performance is the same, or even not as good, as that of *PreInitialAlgo*.

The experiments we conducted also included wall clock time comparison. We present in Table 2 the run time of the experiments presented in Table 1. As we can see from the results presented in Table 2, *PreInitialAlgo* is on average 5 times faster than *DEWPAA*, yet the classification errors of both methods are the same in general, which proves the effectiveness of *PreInitialAlgo*.

The results presented in Table 1 and Table 2 were those for *poolSize* =1000. We conducted other experiments for different values of *poolSize* higher than that, and the results were similar.

An interesting thing to mention is that we computed the wall clock time of *SecOptim;* it took only between 7-12 seconds, which is very small compared to the optimization process of *MainOptim*, so this additional secondary optimization problem we added did not require but a very small additional computational cost, yet the gain was high. (The wall clock time of *SecOptim* is independent of the dataset, since, as we mentioned earlier, *SecOptim* is independent of *MainOptim*)

Table 2. Run time comparison between *PreInitialAlgo* and *DEWPAA* for the experiments presented in Table 1

Dataset	Method	Compression Ratios		
		1:8	1:12	1:16
Lighting7	PreInitialAlgo	00h 19m 59s	00h 13m 20s	00h 09m 34s
	DEWPAA	01h 36m 29s	01h 11m 20s	00h 51m 08s
MedicalImages	PreInitialAlgo	03h 09m 23s	02h 03m 36s	01h 39m 58s
	DEWPAA	16h 35m 59s	11h 25m 40s	08h 48m 33s
Lighting2	PreInitialAlgo	00h 32m 12s	00h 21m 56s	00h 17m 00s
	DEWPAA	02h 30m 13s	01h 31m 27s	01h 16m 06s
MALLAT	PreInitialAlgo	00h 43m 19s	00h 29m 30s	00h 18m 42s
	DEWPAA	03h 30m 42s	02h 08m 55s	01h 37m 23s
FaceUCR	PreInitialAlgo	01h 04m 04s	00h 41m 55s	00h 32m 33s
	DEWPAA	05h 31m 19s	03h 27m 50s	02h 49m 21s
FISH	PreInitialAlgo	02h 57m 03s	02h 47m 01s	01h 53m 45s
	DEWPAA	16h 12m 54s	11h 23m 10s	07h 23m 10s
synthetic_control	PreInitialAlgo	01h 04m 47s	00h 58m 27s	00h 30m 11s
	DEWPAA	05h 56m 10s	04h 01m 09s	03h 12m 27s

5 Conclusion

We presented in this paper a new method, *PreInitialAlgo*, for handling expensive optimization problems such as those encountered in data mining. The new method is applied to population-based bio-inspired algorithms. The basis of our method is to start the optimization process using an optimal initial population. This optimal population is the outcome of another, secondary optimization problem, which is independent of the original problem. We showed experimentally how our new method can substantially improve the performance of the optimization algorithm in terms of speed.

In this paper we used DE and GA as optimizers for the main and secondary optimization problems, respectively. As future work, we would like to test different combinations of bio-inspired algorithms to see which two methods can work best together to yield the best results.

Another direction of future work is to apply the secondary problem using a different fitness function, which could give better results yet.

References

1. Keogh, E., Zhu, Q., Hu, B., Hao. Y., Xi, X., Wei, L. & Ratanamahatana, C.A: The UCR Time Series Classification/Clustering Homepage: www.cs.ucr.edu/~eamonn/time_series_data/ (2011)
2. 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)
3. 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)
4. Muhammad Fuad, M.M.: Genetic Algorithms-Based Symbolic Aggregate Approximation. 14th International Conference on Data Warehousing and Knowledge Discovery - DaWaK 2012 – Vienna, Austria, September 3 – 7. Published in LNCS (2012)
5. 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)
6. 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)
7. 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).
8. Reynolds, C. W.: Flocks, Herds and Schools: A Distributed Behavioral Model. SIGGRAPH Comput. Graph. 21, 4 (1987)