International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2013): 4.438

Comparison between Fast Evolutionary Programming and Artificial Bee Colony Algorithm on Numeric Function Optimization Problems

Mohammad Shafiul Alam[1], Syed Mustafizur Rahman Chowdhury[2], Farhan Al Haque[3], Ridma Hasin[4]

[1, 3, 4]Ahsanullah University of Science and Technology, 141 & 142 Love Road,Tejgaon Industrial Area, Dhaka-1208, Bangladesh

[2]Northern University Bangladesh, House 54, Road 4/A, Dhanmondi, Satmasjid Road, Dhaka-1209, Bangladesh

Abstract: ***The Evolutionary and Swarm Intelligence algorithms are two recently introduced population based meta-heuristic algorithms that have been successfully employed to numerous scientific and engineering problems. In this paper, we have selected two recent and representative algorithms — one from the evolutionary algorithm family, the other from the swarm intelligence family and compared their performance on high dimensional function optimization problems. The evolutionary algorithm that isselected in this paper is the Fast Evolutionary Programming (FEP) which uses Cauchy mutation to improve over the basic Gaussian mutation scheme. The swarm intelligence algorithm that is selected is the Artificial Bee Colony (ABC) algorithm which has been introduced recently and found to be very effective on many continuous optimization problems. This paper compares the performance of these two algorithms on a common set of benchmark problems in order to achieve a better understanding of their algorithmic nature and characteristics. The experimental results show that the performance of ABC is usually better than FEP, especially on complex multimodal functions, because ABC can deal with the problems of premature convergence and fitness stagnation more effectively than FEP.***

Keywords: Evolutionary algorithm, swarm intelligence, fast evolutionary programming, artificial bee colony algorithm,numeric function optimization

1. Introduction

Both the evolutionary algorithms (EAs) and swarm intelligence based algorithms (SIAs) are recently introduced bio-inspired meta-heuristic algorithms that are based on Darwinian theory of evolution and behavior of intelligent swarms found in nature, such as ant colony, bee hive, bird flock, fish school and so on. Since their advent, they have been widely and successfully employed to complex and diverse problems from the fields of science and engineering, such as numeric function optimization [1]–[3], discrete optimization [4], multi-objective optimization [5], industrial process control [6], structural design [7], design of digital IIR filters [8], PID controller [9], machine learning [10] and so on [11]. In comparison to other greedy and local search based algorithms, both EAs and SIAs are more resilient against premature convergence and fitness stagnation, because the population/swarm of candidate solutions can maintain some amount of diversity that is necessary to continue search space explorations avoiding the locally optimal points. However, it is still possible (e.g., [12] – [14]) that the evolving population/swarm of candidate solutions loses its diversity and explorative search capability too soon. This leads the candidate solutions to prematurely get trapped around the local optima of the search space. Aside from premature convergence, another problem faced by both EAs and SIAs is the fitness stagnation, where all the candidate solutions fail to improve their fitness values for indefinitely prolonged iterations, for no apparent reason and even without any premature convergence around the locally optimal points. The risk of premature convergence and fitness stagnation usually rises with reduced explorations and increased exploitations. But, increasing the explorations may lead to unacceptably slow convergence speed. So an adaptive and balanced mix of explorations and exploitations is often necessary for good results and sufficient convergence speed of the algorithm, especially for complex, high dimensional, multimodal objective functions with exponentially many locally optimal points.

To avoid the problems of premature convergence and fitness stagnation, the EA and SIA researchers have developed several improved variants of these algorithms. This paper studies and compares two such improved EA and SIA variants — the Fast Evolutionary Programming (FEP) and the Artificial Bee Colony (ABC) algorithm. Both these algorithms increase their degree of explorations to avoid premature convergence and fitness stagnation. FEP employs a more explorative mutation scheme (i.e., Cauchy mutation), while ABC divides the swarm of candidate solutions and explicitly dedicates some of them (i.e., the ‗employed' bees and ‗scout' bees, as will be explained later in section 5) for more search space explorations. Thus, FEP and ABC try to achieve similar goal (i.e., increased explorations), but using different methodologies (i.e., explorative mutation vs. intelligent division of work). As FEP and ABC belong to different algorithm families (i.e., EA and SIA), they have never been compared side-by-side on the same set of problems, as we have observed through significant literature review. The contribution of this paper is to present an experimental comparison between these two algorithms in order to gain insights on the effectiveness of their different techniques for avoiding premature convergence and fitness stagnation.

The rest of this paper is organized as follows. Section 2 briefly explains the numeric function optimization problem. Section 3 describes the classical evolutionary programming (CEP) which lays the foundation for the Fast Evolutionary Programming (FEP), as explained in the next section.

Section 5 introduces the Artificial Bee Colony (ABC) algorithm in details. Section 6 presents the experimental setup of both FEP and ABC and compares them on seven complex, high dimensional multimodal functions. Finally, section 7 concludes the paper with a brief discussion on both the algorithms and the findings found from their comparison, followed by a few suggestions for future research directions.

2. Numeric Function Optimization Problem

Many real world problems can be formulated as function optimization problems of the parameters that assume values from the continuous domain, i.e., the continuous function optimization problems. A continuous function optimization problem can be formalized as follows.

$$\underset{\boldsymbol{x}}{\text{minimize}} f(\boldsymbol{x}); \quad \text{subject to: } \boldsymbol{x} \in S$$

Here, the goal is to find a vector $\boldsymbol{x}_{\min}$ such that $f(\boldsymbol{x}_{\min}) \leq f(\boldsymbol{x})$ for all $\boldsymbol{x}$ inside S, where the search space S is a bounded subset of $\boldsymbol{R}^n$ and the objective function $f(.)$ is an n dimensional real valued function that is to be optimized over its parameter $\boldsymbol{x}$. Each element $\boldsymbol{x}_i$ of the vector $\boldsymbol{x}$ is a real valued variable: $\boldsymbol{x} = [x_1, x_2, \ldots, x_n]^T$

The task of numeric function optimization is generally referred with many different names, such as real parameter optimization, continuous optimization, or simply, numeric optimization. However, all of them actually refer to the general task of finding a solution across a real valued, (usually) multi-dimensional search space such that the solution gives the best value, i.e., minimum or maximum value, of an objective function, depending on whether it is a minimization or maximization task. This solution should have not only the best objective function value around its local neighborhood, but also the best objective value over all the feasible solutions across the entire search space. In the subsequent sections, we will show how the problem of function optimization is addressed by evolutionary and swarm intelligence algorithms, such as CEP, FEP and ABC.

3. Classical Evolutionary Programming (CEP)

According to the description of Bäck and Schwefel [15], the CEP is implemented as follows.

- Generate an initial population of n individuals. Each individual $\boldsymbol{I}$ is represented as a pair of real valued vectors $(\boldsymbol{x}_i, \boldsymbol{\eta}_i)$, for $i=1, 2, \ldots, n$; Here, $\boldsymbol{x}_i$'s are objective variables and $\boldsymbol{\eta}_i$'s are standard deviations for Gaussian mutations. Each $\boldsymbol{x}_i$ (and $\boldsymbol{\eta}_i$) has D components, where D is the dimensionality of the problem. Each component of $\boldsymbol{x}_i$, for $i=1, 2, \ldots, n$, is generated uniformly at random within its search domain. All the components of $\boldsymbol{\eta}_i$, for $i=1, 2, \ldots, n$, are initialized to some moderate value (e.g., 3.0), as is done in [15].
- Calculate fitness value of each individual $(\boldsymbol{x}_i, \boldsymbol{\eta}_i)$ based on objective function value $f(\boldsymbol{x}_i)$.
- **Mutation step:** Mutate each individual $(\boldsymbol{x}_i, \boldsymbol{\eta}_i)$, for $i=1, 2, \ldots, n$, to create an offspring $(\boldsymbol{x}_i', \boldsymbol{\eta}_i')$ — that is, for $j=1, 2, \ldots, D$,

$$x_i'(j) = x_i(j) + \eta_i(j) N_j(0,1) \quad (1)$$

$$\eta_i'(j) = \eta_i(j) \exp\left(\tau' N_j(0,1) + \tau N_j(0,1)\right) \quad (2)$$

Here, $x_i(j)$, $x_i'(j)$, $\eta_i(j)$, and $\eta_i'(j)$ are the j-th component of the vectors $\boldsymbol{x}_i$, $\boldsymbol{x}_i'$, $\boldsymbol{\eta}_i$ and $\boldsymbol{\eta}_i'$, respectively. $N_j(0, 1)$ is a normally distributed one-dimensional random number with mean=0 and standard deviation=1. Subscript j in $N_j(0,1)$ indicates the random number is generated anew for each j. The factors τ and τ' are set to $\left(\sqrt{2\sqrt{n}}\right)^{-1}$ and $\left(\sqrt{2n}\right)^{-1}$.

- Calculate the fitness of each offspring produced by the previous mutation step.
- Conduct a pair wise tournament based competition over the union of $2n$parents and offspring. For each individual, q opponents are picked uniformly at random from all the other parents and offspring. If the individual's fitness is not less than its opponent in the pair wise competition, it receives a "win".
- Select the n individuals from the union of $2n$parents and offspring that have received the highest number of "win"s. They become the parents for the next generation.
- Stop if some stopping criterion (e.g., predefined maximum number of generations) is fulfilled. Otherwise, go back to the mutation step to start another CEP iteration.

4. Fast Evolutionary Programming (FEP)

The probability density function (PDF) f_t and corresponding cumulative distribution function F_t of one dimensional Cauchy random variable x is given by

$$f_t(x) = \frac{1}{\pi} \frac{t}{t^2 + x^2}, \qquad -\infty < x < \infty \quad (3)$$

$$F_t(x) = \frac{1}{2} + \frac{1}{\pi} \arctan\left(\frac{x}{t}\right) \quad (4)$$

Here, $t > 0$ is the scale parameter. The shape of the PDF $f_t(x)$ is very similar to the Gaussian PDF, but its function plot approaches the x-axis so slowly that the variance of the Cauchy distribution is infinite, and its expected value does not exist. The Fast Evolutionary Programming (FEP) that have been studied for comparison in this paper is exactly the same as the CEP described in the previous section, with the only exception that it employs Cauchy random variable (RV) in eq. (1), instead of the Gaussian RV used by CEP. To be more specific, FEP replaces the eq. (1) of CEP by the following eq. (5).

$$x_i'(j) = x_i(j) + \eta_i(j) \delta_j \quad (5)$$

Since the Cauchy RV has much larger variance (theoretically infinite variance), the Cauchy mutation is more likelyto generate offspring further away from its parents than theGaussian mutations. This ensures higher probability of escaping from a local optimumor moving away from flat plateausand wide basinsof attraction of the fitness landscape. This allows the Cauchy mutation to spend less time in exploitations and more time in explorations, which is why FEP is usually more effective than CEP, especially on more complex multimodal function optimization.

5. Artificial Bee Colony (ABC) Algorithm

Honey bees in a colonyshow remarkable self-organization

and co-ordination skills in their food foraging behavior. Bees have to forage over a vast area in search of good sources of food. After an initial exploration stage, more bees are employed to collect honey from the more profitable food sources whereas fewer bees are assigned to the less worthy food sources. Some scout bees are also assigned for exploration to find newer food sources. If the quality of a food source declines after some exploitation, this information is also shared with other bees so that fewer bees are now attracted to this source. After the quality of a food source falls below some threshold, the bees assigned to it abandon it. The foraging process is initiated by scout bees that start searching for flower patches suitable as food sources. Quality is usually measured as a combination of some values, such as quantity and density of sugar, ease of access, distance from the colony etc. After they return to the hive, those scout bees that found a patch with quality above some threshold, deposit their nectar and then go to the ‗dance floor' to perform a dance known as the ‗waggle dance'. This dance plays the key role to communicate information among the bees about the food sources. The waggle dance contains three pieces of information: i) the quality of the flower patch of this dancing bee, ii) the distance of the flower patch from the hive, iii) the direction from the hive that you have to travel in order to reach the flower. The ‗onlooker' bees, waiting around the dance floor, observe the waggle dances of these ‗employed' bees that have found good food sources and pick any one of them to become its ‗follower' and collect nectar from its flower patch. The better a flower patch as a food source, the bigger is the number of follower bees along with its employed bee. However, if the patch is no longer good enough, it will not be advertised in the next waggle dance and the bees recruited for it as employed or follower bees will choose either to follow some other employed bee or start working as a scout bee to randomly explore the search space for finding new food source.

The ABC algorithm mimics the food foraging behavior of the honey bees with these three groups of bees: employed bees, onlookers and scouts. A bee working to forage a food source (i.e. solution) previously visited by itself and searching only around its vicinity is called an employed bee. Employed bees perform waggle dance to propagate information of its food source to other bees. A bee waiting around the dance floor to choose any of the employed bees to follow is called an onlooker. A bee randomly searching a search space for finding a new food source is called a scout. For every food source, there is only one employed bee and a number of follower bees. The scout bee, after finding a good food source also becomes an employed bee. In the basic ABC algorithm implementation, half of the colony is employed bees and the other half is the onlookers. Number of food sources (i.e., solutions) is equal to the number of employed bees. An employed bee whose food source is exhausted (i.e. solution not improved after several attempts) becomes a scout. The detailed pseudo code is given below.

Step 1) Generate an initial population of N individuals. Each individual is a food source (i.e. solution) and has D attributes, where D is the dimensionality of the problem.

Step 2) Evaluate the fitness of each individual.

Step 3) Each employed beesearches in the neighborhood of its current position to find a better food source. For each employed bee, generate a new solution, $\boldsymbol{v}_i$around its current position, $\boldsymbol{x}_i$ using (6).

$$\boldsymbol{v}_{ij} = \boldsymbol{x}_{ij} + \varphi_{ij} (\boldsymbol{x}_{ij} - \boldsymbol{x}_{kj}) \quad (6)$$

Here, $k \in \{1, 2, \ldots, N_{emp}\}$ and $j \in \{1, 2, \ldots, D\}$ are randomly chosen indices. N_{emp} is the number of employed bees. Φ_{ij} is a uniform random number generated from the range [-1, 1].

Step 4) Compute the fitness of both $\boldsymbol{x}_i$ and $\boldsymbol{v}_i$. Apply greedy selection scheme to choose the betterone and discard the other.

Step 5) Calculate the selection probability, P_i for each solution, $\boldsymbol{x}_i$and normalize the probability value by (7).

$$P_i = fit_i \Big/ \sum_{k=1}^{N} fit_k \quad (7)$$

Step 6) Assign each onlooker bee to a solution, $\boldsymbol{x}_i$ at random with probability proportional to P_i.

Step 7) Produce new food positions (i.e. solutions), $\boldsymbol{v}_i$ for each onlooker bee using the employed bee$\boldsymbol{x}_i$by using (6).

Step 8) Evaluate the fitness of each employed bee, $\boldsymbol{x}_i$and its produced onlooker bee, $\boldsymbol{v}_i$. Apply greedy selection scheme to keep the one with better fitness and discard the other.

Step 9) If a particular solution has not been improved over the past ‗*limit*' cycles (say, *limit* = 100 cycles), then select it for abandonment. Replace it by placing a scout bee at a food source placed uniformly at random over the entire search space by using (8), i.e., for$j = 1, 2, \ldots, D$

$$\boldsymbol{x}_{ij} = min_j + rand(0,1) * (max_j - min_j) \quad (8)$$

Step 10) Keep track of the best solution found so far.

Step 11) Check for termination. If the best solution found is acceptable or maximum number of iterations has elapsed, stop and return the best solution found so far. Otherwise go back to step 2 and repeat.

6. Experimental Studies

To compare the performance of FEP and ABC, we have used a standard benchmark suite of continuous function optimization problems, consisting of sevenhigh dimensional functions [1]–[3], [16]. Table 1 presents a brief overview on each of these benchmark functions. More details on each function can be found in [1]. All these benchmark functions are high dimensional multimodal functions. To optimize a multimodal function, the search algorithm must possess both exploitative and explorative characteristics so that it can explore the locally optimal points without being trapped around any of them. Some of the multimodal functions can have hundreds of local minima, even when the dimensionality is just two or three. Their number of local optima increases exponentially with the number of their dimensions, which makes their optimization extremely difficult. For example, the Ackley function f_3 has one narrow global minimum basin, but with exponentially many minor local minima. The Griewank function f_4 has a component creating linkage among the variables, which complicates the

search by perturbing any subset of the variables. Any technique that tries to optimize each variable separately without considering the others will fail for this function. The difficulty for the Schwefel function f_1 arises from its deep local minima which are far from the single global minimum. All these multimodal functions have exponentially many local minima and the number of local minima increases exponentially with their high dimensionality (i.e., D = 30), making them extremely difficult for any algorithm to be explored and optimized without being trapped around the locally optimal points of the search space.

Table 1: The seven continuous benchmark functions used in our experimental studies. Here, D: dimensionality of the function, S: search space, f_{min}: function value at the global minimum, C: function characteristics with the following values — M: Multimodal, S: Separable, N: Non-separable

No	Function	C	D	S	f_{min}
f_1	Schwefel	MS	30	$[-500, 500]^D$	–12569.5
f_2	Rastrigin	MS	30	$[-5.12, 5.12]^D$	0
f_3	Ackley	MN	30	$[-32, 32]^D$	0
f_4	Griewank	MN	30	$[-600, 600]^D$	0
f_5	Rosenbrock	MS	30	$[-30, 30]^D$	0
f_6	Penalized	MN	30	$[-50, 50]^D$	0
f_7	Penalized2	MN	30	$[-50, 50]^D$	0

Table 2: Performance comparison of FEP and ABC on the benchmark functions. Results are averaged over 50 independent runs. Better performance on each function is marked with boldface font.

No	f_{min}	Generations	FEP		ABC		Better Performance by
			Mean Error	Std. Dev.	Mean Error	Std. Dev.	
f_1	–12569.5	9000	14.9	52.6	**7.28e–11**	**1.44e–11**	ABC
f_2	0	5000	4.6e–02	1.2e–02	**6.12e–16**	**9.30e–17**	ABC
f_3	0	1500	1.8e–02	2.1e–03	**1.22e–11**	**7.10e–12**	ABC
f_4	0	2000	1.6e–02	2.2e–02	**7.31e–16**	**1.32e–16**	ABC
f_5	0	20000	5.06	5.87	**2.77e–02**	**1.88e–02**	ABC
f_6	0	1500	9.2e–06	3.6e–06	**1.22e–11**	**7.09e–12**	ABC
f_7	0	1500	1.6e–04	7.3e–05	**6.95e–16**	**6.12e–17**	ABC

Table 2 presents the results of FEP and ABC on the seven benchmark functions. The common parameter of both the algorithms is the population/swarm size N, which is set to 100. The no. of maximum generations is different for the different functions, as shown in Table 2. The *limit* parameter of ABC is set to 100. All these values are selected after some initial experiments, and not meant to be optimum. Each algorithm has made 50 independent runs on each function. The mean and standard deviation of the error values (i.e., the difference between the global minimum and the best found solution at the final generation) found from the different runs are reported in Table 2. Our observations on the results are summarized in the following few points.

- Out of the seven functions f_1 – f_7, ABC performs consistently better than FEP on all of them. Thus the overall performance of ABC is undoubtedly better than FEP.
- For all these functions, ABC reaches very close to the global minimum value (i.e., mean error ≈ 0), while FEP fails to reach sufficiently close to the global minimum for two functions (i.e., f_1 and f_5). This indicates that FEP fails to satisfactorily deal with the problems of premature convergence and/or fitness stagnation for these functions.
- The performance of ABC is very consistent, i.e., ABC regularly reaches very close to the global minimum, as demonstrated by the very low standard deviation values of its errors on the different functions.

In summary, ABC is more effective than FEP on the complex, multimodal functions. The reason might be that FEP performs excessive explorations all through its execution by using the Cauchy mutation scheme. As the Cauchy distribution has very large (theoretically, infinite) variance, FEP may not converge and may not provide the necessary exploitation which is usually required for fine tuning during the final phase of any optimization. On the other hand, ABC achieves a good balance between exploitations and explorations by using equal number of employed and onlooker bees, as demonstrated by its excellent results on all the benchmark functions.

7. Conclusion

This paper compares an evolutionary family algorithm (i.e., FEP) with a swarm intelligence algorithm (i.e., ABC) using several benchmark functions. These two algorithms have never been compared side-by-side on the same problem set, which is the main contribution of this paper. Results indicate that ABC can perform better than FEP on most of the multimodal functions. There might be several possible future research directions based on FEP and ABC. Firstly, they should be compared on easier unimodal and low dimensional functions to gain further insights on their characteristics. Secondly, since FEP is more explorative than ABC, it might be possible to hybridize them together into a new, single algorithm. This new algorithm may deploy FEP during the early generations when more explorations are desired, but later may switch to more exploitations by employing ABC.Thirdly, the possibility of improving the final solution quality might be investigated by using an efficient local searcher after the execution of both FEP and ABC is over.

This may make their performance closer. Finally, both FEP and ABC are applied only on the benchmark functions. It would be interesting to study how well they can perform on many other existing problems, especially the discrete and real world ones.

References

[1] D. Karaboga and B. Basturk, On the performance of artificial bee colony (ABC) algorithm, Applied Soft Computing8 (1) (2008) 687–697.
[2] D. Karaboga, An idea based on honey bee swarm for numerical optimization, Erciyes University, Kayseri, Turkey, *Technical Report-TR06*, 2005.
[3] X. Yao, Y. Liu and G. Lin, ―Evolutionary programming made faster", IEEE Transactions on Evolutionary Computation**3** (2) (1999) 82–102.
[4] S. Sobti and P. Singla, Solving travelling salesman problem using bee colony based approach, *International Journal of Engineering Research and Technology***2** (6) (2013) 186–189.
[5] K. Naidu, H. Mokhlis and A.H.A. Bakar, Multiobjective optimization using weighted sum Artificial Bee Colony algorithm for Load Frequency Control, *International Journal of Electrical Power and Energy Systems***55** (2) (2014) 657–667.
[6] R. Mukherjee, D. Goswami and S. Chakraborty, Parametric optimization of Nd:YAG laser beam machining process using artificial bee colony algorithm, *Journal of Industrial Engineering*, vol. 2013, Article ID 570250, 15 pages, 2013. DOI: 10.1155/2013/570250.
[7] H. Garg, Solving structural engineering design optimization problems using an artificial bee colony algorithm, *Journal of Industrial and Management Optimization*,**10** (3) (2014) 777–794.
[8] Z. Zhao, D. Yin and Y. Jiang, Improved bee colony algorithm based on knowledge strategy for digital filter design, *International Journal of Computer Applications*,**47** (2) (2013) 241–248.
[9] A. Mishra, A. Khanna, N. Singh and V. Mishra, Speed control of DC motor using bee colony optimization, *Universal Journal of Electrical and Electronic Engineering* **1** (3) (2013) 68–75.
[10] A. Karegowda and M. Darshan, Optimizing feed forward neural network connection weights using artificial bee colony algorithm, *International Journal of Advanced Research in Computer Science and Software Engineering* **3** (7) (2013) 452–454.
[11] A. Bolaji, A. Khader, M. Betar and M. Awadallah, Bee colony algorithm, its variants and applications: A survey, *Journal of Theoretical and Applied Technology* **47** (2)(2013) 434–459.
[12] T. Park and K. R. Ryu, A Dual population genetic algorithm for adaptive diversity control, *IEEE Trans. Evolutionary Computation* **14** (6) (2010) 865–884.
[13] R. K. Ursem, Diversity guided evolutionary algorithms, in *Proc. 7th Int. Conf. Parallel Problem Solving from Nature* (*PPSN*), 2002, pp. 462–474.
[14] J. Lampinen and I. Zelinka, On stagnation of the differential evolution algorithm, in *Proc. 6th Int. Mendel Conf. Soft Computing*, Brno, Czech Republic, 2000, pp. 76–83.
[15] T. Bäck and H.–P.Schwefel, ―An overview of evolutionary algorithms for parameter optimization", Evolutionary Computation **1** (1) (1993) 1–23.
[16] W. Lee and W. Cai, A novel artificial bee colony algorithm with diversity strategy, in *Proc. 7th Int. Conf. Natural Computation*, 2011, pp. 1441–1444.