International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2015): 78.96 | Impact Factor (2015): 6.391

A Comparative Study of Classical Evolutionary Programming and Bat Algorithm

D. M. Anisuzzaman[1], Shifat Sharmin Shapla[2]

[1]Ahsanullah University of Science and Technology, 141 &142 Love Road, Dhaka-1208, Bangladesh

[2]Stamford University Bangladesh, 51 Siddeswari Road, Dhaka-1217, Bangladesh

Abstract: ***Evolutionary Programming is an algorithm which uses natural selection of the fittest. In Classical Evolutionary Programming (CEP) uses Gaussian mutation. CEP is better at search in a small local neighborhood. It offers the best results for the unimodal and multimodal functions with only a few local minima. On the other hand, Swarm Intelligence algorithm inspired by natural behavior. Swarm Intelligence algorithm is the collective behavior of decentralized self-organized system. In this paper, we conducted an experimental comparison between CEP and a Swarm Intelligence algorithm ─ BAT. The experimental data shows that CEP performs better than BAT algorithm.***

Keywords: Classical Evolutionary Programming, Swarm Intelligence Algorithm, BAT algorithm, offspring, continuous optimization

1. Introduction

Evolutionary Programming is a global optimization algorithm and inspired by the theory of evolution by means of natural selection. Evolutionary programming which uses Gaussian mutation is known as Classical Evolutionary Programming(CEP). CEP is better at search in a small local neighborhood. It offers the best results for the unimodal and multimodal functions with only a few local minima. Swarm Intelligence (SI) is the collective behavior of self-organized system, natural or artificial. Bat algorithm is a recent swarm intelligence algorithm based on the echolocation behavior of microbats. In this paper an experimental comparison between CEP and BAT algorithm have done. The rest of the paper is organized as —section 2 and 3 describe the CEP and BAT algorithm respectively, section 4 represents the experimental data and finally a discussion and findings are shown in the section 5.

2. Classical Evolutionary Programming (CEP) Algorithm

Classical Evolutionary Programming (CEP) is a kind of Evolutionary Programming which uses Gaussian mutation. Evolutionary programming (EP) has recently been applied to many combinatorial and numerical optimization problems [1][2], though it was first proposed as an approach to artificial intelligence [3]. Optimization by EP can be summarized into two major steps:

1) mutate the solutions in the current population;
2) select the next generation from the mutated and the current solutions.

They can be considered as a population-based version of the classical generate-and-test method [4], where new solutions (offspring) are generated from the mutation and which of the newly generated solutions should survive to the next generation is tested by the selection.

2.1 Function Optimization by CEP

A global minimization problem can be formalized as a pair (S,f), where S $\subset R_n$ is a bounded set on Rn and f : S → R is an n-dimensional real-valued function. The problem is to find a point x_{min} such that $f(x_{min})$ is a global minimum on S. [5]. Fogel [6][7] and Bäckand Schwefel [8] have indicated that CEP without self-adaptive mutation usually performs worse than CEP with self-adaptive mutation for the functions they tested. Hence the CEP with self-adaptive mutation will be investigated here.

2.2 Pseudo Code of CEP Algorithm

According to the description by Bäck and Schwefel [8], the CEP is implemented as follows:

1) Generate the initial population of μ individuals, and set k=1. Each individual is taken as a pair of real valued vectors, (x_i, η_i), $\forall_i \in \{1, \ldots, \mu\}$, where x_i's are objective variables and η_i's are standard deviations for Gaussian mutations.
2) Evaluate the fitness score for each individual (x_i, η_i), $\forall_i \in \{1, \ldots, \mu\}$, of the population based on the objective function, $f(x_i)$.
3) Each parent (x_i, η_i), i = 1, ….. , μ, creates a single offspring $(xi', \eta i')$.
4) Calculate the fitness of each offspring (x_i', η_i'), $\forall_i \in \{1, \ldots, \mu\}$.
5) Conduct pairwise comparison over the union of parents (x_i, η_i) and offspring (x_i', η_i'), $\forall_i \in \{1, \ldots, \mu\}$. For each individual, q opponents are chosen uniformly at random from all the parents and offspring. For each comparison, if the individual's fitness is no smaller than the opponent's, it receives a "win."
6) Select the μ individuals out of (x_i, η_i) and (x_i', η_i'), $\forall_i \in \{1, \ldots, \mu\}$, that have the most wins to be parents of the next generation.
7) Stop if the halting criterion is satisfied; otherwise, k = k + 1 and go to Step 3.

3. Bat Algorithm

Bat algorithm was inspired by the echolocation behavior of microbats. It is a meta-heuristic algorithm with varying pulse rates of emission and loudness. [9]. Bats are born with the capability of echolocation. Bats emit a high sound frequency that bounces back from the neighboring objects. They listen to this echo to find their way [10][11]. Depending on their species the signal frequencies varies [12][13]. Microbats echolocation characteristics highlight on some approximate rules. They are given below [14].

1) **Distance:** Bats sense distance by using echolocation. They acknowledge the ranges/spaces between prey and surrounded barriers.
2) **Frequency:** Bats fly randomly with velocity vi at position xi with a fixed frequency f_{min}, wavelength λ and loudness A0 to search for prey. They can automatically adjust the wave length of their emitted pulses and adjust the rate of pulse emission in the range of [0, 1], depending on the proximity of their target [15].
3) **Loudness:** Though loudness can vary in many ways. Here assume that the loudness differs from a large A0 to a minimum constant value A_{min}.

3.1 Initialization of Bat Algorithm

The initial population is generated randomly. For generating the initial population each individual of the population formed of real valued vectors with dimension D and n number of bats.

$$x_{ij} = x_{minj} + rand\ 0{,}1\ (\ x_{\max j} - x_{minj}\ ) \quad (1)$$

Where i = 1,2, ……., n; j = 1,2, ……, D; $x_{\max j}$ and $x_{\min j}$ are the upper and lower boundaries for dimension.

3.2 Solution, Frequency and Velocity

In Bat algorithm step size of a new solution is defined by the frequency. For each solution, the frequency is set to a value which ranges between upper and lower boundaries f_{min} and f_{max}. The bat position and velocity is updated by using the frequency. Following equations are used to update velocity and positions[16].

$$f_i = f_{min} + (f_{max} - f_{mi_n})\ \beta \quad (2)$$
$$V_i{}^t = V_i{}^{t-1} + x_i^t - x^*)\ f_i \quad (3)$$
$$x_i{}^t = x_i{}^t + V_i{}^t \quad (4)$$

Where $\beta \in$ [0, 1] indicates randomly generated number, x* represents the global best solutions in the population, fi is the frequency for the solution i, Vi represents the new velocity for the solution i. A solution is selected among the best solution and random walk is applied in order to increase exploration. Thus a new candidate solution is generated.

$$x_{new} = x_{old} + \overline{\varepsilon At} \quad (5)$$

$\overline{At}$ is the average loudness of all the bats and $\varepsilon \epsilon$ [0, 1] is uniform random number that represents the directions and intensity of random walk.

3.3 Loudness and Pulse Emission Rate

As iterations proceed loudness and pulse emission rate must be adjusted and updated. Getting closer to its prey, bat usually decrease loudness A and increases the pulse emission rate [17][18]. Loudness A and pulse emission rate r are updated by using following equations:

$$A_i{}^{t=1} = \alpha A_i{}^t \quad (6)$$
$$r_i{}^{t+1} = r_i{}^0\ [1 - (-\gamma t)] \quad (7)$$

Where α and γ are constants which having the determined values for these equations which is set to 0.9 in this simulation. Here the initial loudness $A_i{}^0$ can typically be [1, 2], while the initial emission rate $r_i{}^0$ can be [0, 1] normally [19].

3.4 Pseudo Code of CEP Algorithm

The initial population is generated randomly. Each individual of t
1. Objective function: $f\ (x)$, x = ($x1$, $x2$, $x3$, … … … . , $xd)T$
2. Initialize bat population xi and velocity v ; i = (1,2, … … … , n)
3. Define pulse frequency $fiatxi$
4. Initialize pulse rate ri and loudness Ai
5. while (t< maximum number of iterations)
6. Generate new solutions by adjusting frequency,
7. and updating velocities and locations/solutions
8. if (rand >ri)
9. Select a solution among the best solutions
10. Generate a local solution around the selected best solution
11. end if
12. if (rand <Ai) **and** $f\ (xi)$ <$f\ (x*)$
13. Accept new solutions
14. increase ri, reduce Ai
15. end if
16. Rank the bats and find current best $x*$
17. end while
18. Display results.

4. Simulation and Experimental Result

To compare the performance of CEP and BAT algorithm, we have used seven standard benchmark functions with the dimension, D = 30 including both unimodal and multimodal functions. The analytical form of each functions with global minimum values are shown in the following Table 1. These functions are tested for both the algorithms. Here, the experimental results of CEP and BAT algorithms for seven benchmark functions are shown in Table 2. The maximum generations vary from function to function. The mean and standard deviation of the error values found from the simulation are shown in Table 2. The overview of the result is summarized in the following points.

- CEP performs better than BAT algorithm for all of the seven functions.
- For all these seven functions, CEP reaches very close to the global minimum value, while the BAT algorithm fails to reach.

Finally, CEP is far better than Bat algorithm on both unimodal and multimodal functions. The reason might be

Cauchy mutation performs better because of its higher probability of making longer jumps and CEP uses Cauchy mutation for creating new offspring, while the Bat algorithm usually gets trapped around local minima which is far from the global minimum.

Table 1: Benchmark functions used in the experimental studies. Here, D: Dimensionality of the function, S: search space, C: function characteristics with values — U: Unimodal and M: Multimodal.

No.	Name	D	C	S	Function	f_{min}
f_1	Sphere	30	U	$[-5.12, 5.12]^D$	$f_1(x) = \sum_{i=1}^{D} X_i^2$	0
f_2	Step	30	U	$[-100, 100]^D$	$f_2(x) = \sum_{i=1}^{d} (\lfloor x_i + 0.5 \rfloor^2)$	0
f_3	Rosenbrock	30	U	$[-15, 15]^D$	$f_3(x) = \sum_{i=1}^{d-1} [100\,(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	0
f_4	Quartic	30	U	$[-1.28, 1.28]^D$	$f_4(x) = \sum_{i=1}^{n} i x_i^4$	0
f_5	Griewank	30	M	$[-600, 600]^D$	$f_5 = \frac{1}{4000}\sum_{i=1}^{d} x_i^2 - \prod_{i=1}^{d} cos\frac{x_i}{\sqrt{i}} + 1$	0
f_6	Rastrigin	30	M	$[-15, 15]^D$	$f_6(x) = \sum_{i=1}^{d} [x_i^2 - 10\cos(2\pi x_i) + 10]$	0
f_7	Ackley	30	M	$[-32, 32]^D$	$f_7(x) = -20\exp(-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}) - \exp(\frac{1}{n}\sum_{i=1}^{n} \cos 2\pi x_i) + 20 + e$	0

Table 2: Performance comparison of ABC and Bat algorithm on the benchmark functions. Better performance on each function is marked with boldface font

No.	f_{min}	CEP		BAT		Batter Performance by
		Mean	Std. Dev.	Mean	Std. Dev.	
f_1	0.0	2.20E-04	5.90E-4	2.48E+01	6.75E+00	**CEP**
f_2	0.0	577.76E+00	1125.76E+00	1.18E+04	3.53E+03	**CEP**
f_3	0.0	6.17 E+00	13.61E+00	4.39E+05	3.24E+05	**CEP**
f_4	0.0	1.80E-02	6.41E-03	1.29E+01	2.21E+00	**CEP**
f_5	0.0	0.14 E+00	0.12E+00	9.19E+01	2.48E+01	**CEP**
f_6	0.0	4.08 E+00	3.08E+00	4.54E+02	7.81E+01	**CEP**
f_7	0.0	8.1E-02	0.34E+00	1.62E+01	1.14E+00	**CEP**

5. Conclusion

The This paper shows a comparative study between a evolutionary programming algorithm and a swarm intelligence based algorithms — the Classical Evolutionary Programming (CEP) algorithm and BAT algorithm. The CEP algorithm performs better than the Bat algorithm on all the seven benchmark functions used in our study. The reason behind this might be that — CEP uses Cauchy mutation, which creates longer jump that prevents premature convergence around the locally optimal points and helps find near optimal solutions. On the other hand, the Bat algorithm fails to find global optimum solution on most of the functions as it converses around the locally optimal points.

References

[1] D. B. Fogel, System Identification Through Simulated Evolution: A Machine Learning Approach to Modeling. Needham Heights, MA: Ginn, 1991.
[2] "Applying evolutionary programming to selected traveling salesman problems," *Cybern. Syst.*, vol. 24, pp. 27–36, 1993.
[3] L. J. Fogel, A. J. Owens, and M. J. Walsh, *Artificial Intelligence Through Simulated Evolution*. New York: Wiley, 1966.
[4] X. Yao, "An overview of evolutionary computation," *Chinese J. Adv. Software Res.*, vol. 3, no. 1, pp. 12–29, 1996.
[5] D. B. Fogel, "An introduction to simulated evolutionary optimization," *IEEE Trans. Neural Networks*, vol. 5, pp. 3–14, Jan. 1994.
[6] "Evolving artificial intelligence," Ph.D. dissertation, Univ. of California, San Diego, CA, 1992.
[7] D. B. Fogel, Evolutionary Computation: Toward a New Philosophy of Machine Intelligence. Piscataway, NJ: IEEE Press, 1995.
[8] T. B¨ack and H.-P. Schwefel, "An overview of evolutionary algorithms for parameter optimization," *Evol. Comput.*, vol. 1, no. 1, pp. 1–23, 1993.
[9] J. D. Altringham, —Bats: Biology and Behaviourl, Oxford University Press, (1996).
[10] Y. Selim and U. K. Ecir, "Improved Bat Algorithm (IBA) on Continuous Optimization Problems," Lecture Notes on Software Engineering, vol. 1, no. 3, pp. 279-283, 2013.
[11] T. Colin, "The Varienty of Life", Oxford University Press, 2000.

[12] A. Faritha and C. Chandrasekar, "An optimized approach of modified bat algorithm to record de duplication," International Journal of Computer Applications, vol. 62, no. 1, pp. 10-15, 2012.
[13] Y. Xin-She, "Bat Algorithm for Multiobjective Optimization," International Journal Bio-Inspired Computation, vol. 3, no. 5, pp. 267-274, 2011.
[14] Y. Xin-She, "A New Metaheuristic Bat-Inspired Algorithm, Nature Inspired Cooperative Strategies for Optimization (NISCO)", Springer, vol. 284, no. Springer Berlin, pp. 65-74, 2010.
[15] Y. Xin-She, "Bat Algorithm for Multiobjective Optimization," International Journal Bio-Inspired Computation, vol. 3, no. 5, pp. 267-274, 2011.
[16] N. Sakib, S. Mustafizur, M. S. Alam, M. W. U. Kabir, "A Novel Adaptive Bat Algorithm to Control Explorations and Exploitations for Continuous Optimization Problems," International Journal of Computer Applications, vol. 94, no. 13, 2014.
[17] Y.Xin-She, "A New Metaheuristic Bat-Inspired Algorithm, Nature Inspired Cooperative Strategies for Optimization (NISCO 2010)," Springer, vol. 284, no. Springer Berlin, pp. 65-74 , 2010.
[18] Y. Xin-She, "Bat Algorithm for Multiobjective Optimization," International Journal Bio- Inspired Computation, vol. 3, no. 5, pp. 267-274, 2011.
[19] X. S. Yang, "Harmony Search as a Metaheuristic Algorithm, Music-Inspired Harmony Search Algorithm," Theory and Applications, Studies in Computational Intelligence, vol. 191, pp. 1-14, 2009.