μABC: A Micro Artificial Bee Colony Algorithm for Large Scale Global Optimization

Anguluri Rajasekhar
Dept of Electrical and Electronics Engg
National Institute of Technology, Warangal, INDIA
(+91) 9705457670
rajasekhar.anguluri@ieee.org

Swagatam Das
Electronics and Communication Science Unit
Indian Statistical Institute, Kolkata
(+91) 9831219774
swagatam.das@ieee.org

Sanjoy Das
Electrical and Computer Engineering,
Kansas State University,
Kansas, Manhattan, USS
001-785-585-4642
sdas@ksu.edu

ABSTRACT

In this paper, we propose a new variant of Artificial Bee Colony Algorithm termed as μABC: Micro Artificial Bee Colony algorithm, which evolves with a very small population compared to its traditional version. In this approach the bees are ranked via their fitness. Best bee is kept unaltered, whereas the other bees are reinitialized with help of some modifications based on the food source obtained by best bee. This type of raking system will always help bees (apart from best bee) to exploit areas in the vicinity of food source corresponding to best bee. μABC is validated over a benchmark suite of shifted functions suggested in CEC'2008 competition and compared with the methods like EPS-PSO, CCPSO2, etc. Various comparisons with dimensions greater than 100 show the performance of μABC in solving higher dimensional problems with less computational effort.

Categories and Subject Descriptors

D.3.3 [**Artificial Intelligence**]: Numerical Function Optimization.

General Terms

Algorithms

Keywords

μABC, Rechenberg's rule

1. INTRODUCTION

Artificial Bee Colony (ABC) is one of the most recent, swarm intelligence based algorithms employing the foraging behavior of honey bees for solving dynamic numerical optimization problems [1]. Most successful, methods that are proposed for large scale optimization include modification of search equation and also on mutation strategies, Akay and karaboga [2] provided more control parameters rather than single control *limit* in basic ABC.

2. μABC: MICRO ARTIFICIAL BEE COLONY ALGORITHM

A population of three bees' evolves through iterations in this proposed μABC framework. After one cycle (after employed and onlooker phase has been completed) bees in population move to some new foraging locations. At this point, the population may be ranked according to Fitness values. The best bee (rank 1) of the population retains its position. The second best bee (rank 2) is moved to a position very close to the best one. Worst bee (rank 3) is initialized to a random position. After each complete cycle best bee is expected to contain the most valuable information about the fitness landscape and thus, by retaining its position we can conserve the best location so far discovered. Now, the second best bee is drawn to a position close to the best bee in order to facilitate local search during next loop execution. The worst bee is utilized in maintaining population diversity and avoiding premature convergence. As the population size is very small, scout bee exploration is not carried out as this task is been guided by the worst bee. As μABC is composed of small number of bees meticulous care should be taken in generating the new solutions. For this purpose the following operations and mechanisms are incorporated. Fig 1 shows the relocation of bees after one complete cycle.

(i) Control of the number of variables to be perturbed:

In the basic version of ABC, only one randomly chosen variable is perturbed while the others are copied form the old solution. The number of variables being perturbed in the current solution is called and it is kept fixed in conventional ABC resulting into slow convergence rate [2]. This problem is addressed by introducing a control parameter known as Frequency Control Rate (*FCR*). By means of this modification, for each parameter x_{ij} a uniformly distributed random number $rand_{i,j}(0,1)$, is instantiated and if the random number is less than the current value of *FCR* then the variable x_{ij} is modified in the following way:

$$v_{ij} = \begin{cases} x_{ij} + \phi_{ij}.(x_{ij} - x_{kj}), & \text{if } rand_{i,j}(0,1) <= FCR \\ x_{ij} & \text{Otherwise} \end{cases} \quad (1)$$

where $k \in \{1,2,....,FS\}$ is a randomly chosen index that has to be different from *i*. Random values of *FCR* may or may not result into good performance as lower value of *FCR* cause solutions to improve slowly while a higher one may cause too much diversity in a solution and hence in the population.

(ii) Controlling the range of perturbation:

Our experimental investigations indicate that fixing the bounds of the scaling factor $\varphi_{i,j}$ of Eqn. (1) to constant values (-1 and 1), as in done for classical ABC, results in poorer exploration of the high-dimensional search space. In μABC, $\phi_{i,j}$ is constricted in the range [*-RF, RF*] where *RF* is termed as the Range Factor and its values are adaptively updated with the progress of the

GECCO'12 Companion, July 7–11, 2012, Philadelphia, PA, USA.
ACM 978-1-4503-1178-6/12/07.

algorithm. Note that the lower value of *RF* allows the search to fine tune the process in small steps leading to slow convergence while larger value of *RF* enhances the searching speed at the same time exploitation capability of the algorithm reduces. Hence the magnitude of perturbation is controlled by a control parameter called the Ratio factor *RF*. This value is set before running the algorithm.

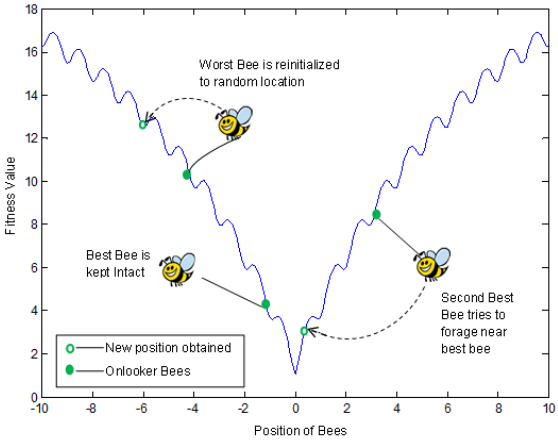

Fig. 1 Bees are relocated based on their ranks

To balance the level of searching speeds µABC is been designed to change *RF* automatically during the search. To conduct automating tuning of *RF* Rechenberg's 1/5 mutation rule is being used. According to this rule the ratio of successful mutation to all mutations should be 1/5. Changing the step size according to 1/5 rule in every *m* cycles is done according to Eqn (2). If the algorithm cannot improve the solution with respect to Rechenberg 1/5 rule, that is the ratio of successful mutations $\varphi(m)$ is less than 1/5, *RF* is decreased. If $\varphi(m)$ is greater than 1/5 then *RF* is increased in order to speed up the search.

$$RF_{new}(t+1) = \begin{cases} RF_{old}(t) * 0.85 & \text{if } \varphi(m) < 1/5, \\ RF_{old}(t) / 0.85 & \text{if } \varphi(m) > 1/5 \\ RF_{old}(t) & \text{if } \varphi(m) = 1/5 \end{cases} \quad (2)$$

3. EXPERIMENTAL RESULTS

The proposed algorithm has been tested on 7 shifted benchmark functions suggested in CEC'2008 and results are recorded and compared in Table 1, 2 for 100, 500-Dimensions. The value of *FCR* is fixed to 0.4. From the empirical results it is very clear that µABC had outperformed three state-of-art variants of PSO, DE.

4. REFERENCES

[1] Karaboga, D., and Basturk, S. 2007. A powerful and efficient algorithm for numerical optimization: Artificial Bee Colony (ABC) Algorithm. *J of Global Optim,* vol 39, issue 3, 459-471.

[2] Bahriye A, Karaboga D. 2012. A modified Artificial Bee Colony Algorithm for real-parameter optimization, *Information Sciences,* vol 192, pp. 120-142.

[3] Li, X. and Yao, X. 2011. Cooperatively Coevolving Particle Swarms for Large Scale Optimization. *IEEE Trans on Evolutionary Comp,* doi: 10.1109/TEVC.2011.2112662.

[4] Hsieh, T,S. Sun, T,Y. Liu, C,C. and Tsai, S,J. 2008. Solving large scale global optimization using improved Particle Swarm Optimizer, *CEC'2008 (IEEE World Congress on Computational Intelligence),* Hong Kong, 1777-1784.

[5] Zamuda, A. Brest, J. Boskovic, B. and Zumer, V. 2008. Large Scale Global Optimization using Differential Evolution with self-adaption and cooperative co-evolution, *CEC'2008 (IEEE World Congress on Computational Intelligence),* Hong Kong, 3718-3725.

Table 1: Average Error and Standard deviation of the 25 independent runs tested on CEC'2008 benchmark (D=100)

Function	F_1	F_2	F_4	F_5	F_6	F_7
µABC	**2.24e-15 (1.15e-16)**	**5.20e+00 (8.90e-01)**	**0.00e+00 (0.00e+00)**	**4.07e-16 (2.56e-16)**	**1.10e-13 (1.36e-14)**	**-1.54e+03 (3.91e+00)**
CCPSO2 [3]	7.73e-14 (3.23e-14)	6.08e+00 (7.83e+00)	3.98e-02 (1.99e-01)	3.45e-03 (4.88e-03)	1.44e-13 (3.06e-14)	-1.50e+03 (1.04e+01)
EPUS-PSO [4]	7.47e-01 (1.70e-01)	1.86e+01 (2.26e+00)	4.71e+02 (5.94e+01)	3.72e-01 (5.60e-02)	2.06e+00 (4.40e-01)	-8.55e+02 (1.35e+01)
DEwSAcc[5]	5.68e-14 (0.00e+00)	8.25e+00 (5.32e+00)	4.37e+00 (7.65e+00)	3.06e-14 (7.86e-15)	1.12e-13 (1.53e-14)	-1.36e+03 (2.45e+01)

Table 2: Average Error and Standard deviation of the 25 independent runs tested on CEC'2008 benchmark (D=500)

Function	F_1	F_2	F_4	F_5	F_6	F_7
µABC	**9.64e-15 (2.35e-16)**	**5.06e+01 (1.42e+00)**	3.64e+01 (7.37e+00)	**4.16e-04 (7.20e-04)**	**4.27e-13 (1.56e-14)**	**-7.45e+03 (2.53e+01)**
CCPSO2 [3]	3.00e-13 (7.96e-14)	5.79e+01 (4.21e+01)	**3.98e-02 (1.99e-01)**	1.18e-03 (4.61e-03)	5.34e-13 (8.61e-14)	-7.23e+03 (4.16e+01)
EPUS-PSO [4]	8.45e+01 (6.40e+00)	4.35e+01 (5.51e-01)	3.49e+03 (1.12e+02)	1.64e+00 (4.69e-02)	6.64e+00 (4.49e-01)	-3.51e+03 (2.10e+01)
DEwSAcc[5]	2.09e-09 (4.61e-09)	7.57e+01 (3.04e+00)	3.64e+02 (5.23e+01)	6.90e-04 (2.41e-03)	4.80e-01 (5.73e-01)	-5.74e+03 (1.83e+02)