Applied Soft Computing 38 (2016) 1040–1048

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Self organizing migrating algorithm with quadratic interpolation for solving large scale global optimization problems

Dipti Singh[a], Seema Agrawal[b,*]

[a] Department of Applied Sciences, Gautam Buddha University, Greater Noida, India
[b] Department of Mathematics, S.S.V. (P.G.) College, Hapur, C. C. S. University, India

A R T I C L E I N F O

Article history:
Received 30 November 2014
Received in revised form 24 June 2015
Accepted 24 September 2015
Available online 9 October 2015

Keywords:
Self organizing migrating algorithm
Quadratic interpolation
Large scale global optimization

A B S T R A C T

Generally the complexity of the large scale optimization problem is considered to increase as the size or dimension of the problem increases and to solve these problems; more efficient and robust algorithms are needed. Several experiments have shown that an increment in dimensions of the problem not only requires an increment in population size but increases the computational cost also. In this paper a Self Organizing Migrating Algorithm with Quadratic Interpolation (SOMAQI) has been extended to solve large scale global optimization problems for dimensions ranging from 100 to 3000 with a constant population size of 10 only. It produces high quality optimal solution with very low computational cost and converges very fast to optimal solution.

1. Introduction

Large scale optimization problems arise in many real world applications. With the advent of computers, high speed computational algorithms for solving these problems are in demand. Due to their wide applicability and easy implementation, population based stochastic techniques are considered as better option. But, the main drawback of using these techniques is that they require a reasonable amount of population size with an increment in dimension which in turn incurs high computational cost in terms of function evaluations as well as run time. Many approaches have been proposed in past to solve large scale optimization problems. Ahmed Fouad Ali [1] proposed a hybrid simulated annealing and Nelder–Mead algorithm (SNMRUV) for solving large scale global optimization problems. Its performance was tested on 27 scalable benchmark functions. Among these functions 10 classical functions were tested for 16–512 dimensions only and 17 hard functions were tested for 500 and 1000 dimensions over 30 runs and the algorithm performed well for some functions but not for all. Sheng-Ta Hsieh et al. [2] presented the Efficient Population Utilization Strategy for Particle Swarm Optimizer (EPUS-PSO) and tested its performance on the seven test functions with 100, 500 and 1000 dimensions respectively over 25 runs only and the results achieved for all the problems were not so good. Kazimipour [3] categorized the most powerful initialization method among the earlier developed population based initialization methods and tested its performance on CEC'2008 benchmark functions with 100, 500 and 1000 dimensions for 50 independent runs. The population size was kept constant (i.e. 50) for all initialization methods and it was concluded that QBL could be considered the best method among the eight initialization methods. Rajasekhar et al. [4] provided a new variant of Artificial Bee Colony Algorithm (μABC) for large scale global optimization and tested its performance on 7 shifted benchmark functions given in CEC'2008 with 100 and 500 dimensions for 25 independent runs. Its performance was compared with other algorithms and was found that the algorithm performed well for all the problems except one. Antonio LaTorre et al. [5] selected best hybrid algorithm among the combinations of several well known optimization algorithms and its performance was tested on 15 benchmark functions proposed for the special session on large scale global optimization and compared with the reference algorithm, different population sizes were used for different hybrid algorithms (25–400) and the algorithm performed well for all the problems except two. A modified line search method proposed by Grosan et al. [6] makes use of partial derivatives and restarts the search process after a given number of iterations. Its performance has been tested on a set of 8 standard continuous test functions with dimensions 50–10,000 and compared with Genetic Algorithm (GA) and Particle Swarm Optimization (PSO). Population size has been taken very large as 500. Rahnamayan et al. [7] investigated Opposition Based Differential Evolution (ODE) based on Differential Evolution and tested it on seven benchmark functions with 500 and 1000 dimensions. Population size for the proposed work was taken same as the dimensions and as a result computational cost

* Corresponding author. Tel.: +91 9540119292; fax: +91 122 2306645.
E-mail address: seemagrwl7@gmail.com (S. Agrawal).

http://dx.doi.org/10.1016/j.asoc.2015.09.033
1568-4946/© 2015 Elsevier B.V. All rights reserved.

was very high. Rajasekhar et al. [8] introduced an improved version of Artificial Bee Colony (ABC) algorithm with mutation based on Levy Probability Distributions and tested its performance on 7 standard benchmark functions with 100 dimensions and on a set of non-traditional problems with 500 dimensions suggested in the special session of CEC'2008. Population size was taken as 20 for 100 dimensions problems and 50 for 500 dimensions problems and the results obtained were quiet good for some functions. Literature available for large scale optimization shows that many of the efficient algorithms not only require large population size but also require large computation cost. This paper focuses over all above mentioned issues and proposed an alternative approach for solving large scale problems.

SOMAQI has been developed earlier to solve unconstrained optimization problems of dimension up to 50 only. In this paper, SOMAQI has been extended to solve problems of dimension up to 3000 keeping a constant population size of 10. An experiment has been made to fine tune the population size by varying it from 10 to 100. The novelty of this approach is that where other algorithms use population size of 2 to 3 times of dimension size, SOMAQI is using only population size 10 and providing good success rate with high quality solutions. Validity of SOMAQI for large scale problems has been tested on 12 benchmark test functions. Numerical results show that SOMAQI is robust algorithm which gives quality solution of large scale problems within seconds in less number of function evaluations.

The paper is organized as follows. In Section 2, SOMA is described. In Section 3, the methodology of SOMAQI has been presented. The performance of SOMAQI for solving large scale problems has been discussed in Section 4. Finally, the paper concludes with Section 5.

2. Self organizing migrating algorithm

Self Organizing Migrating Algorithm is a population based stochastic optimization technique modeled on the social behavior of a group of individuals [9]. Rather than competing with each other, individuals in SOMA follow the competitive–cooperative behavior. It is classified as an Evolutionary Algorithm (EA), but in contrast of EA, SOMA does not create any new individual during the search; only the positions of the individuals are changed during a generation called "migration loop" (ML). Like other EAs, its working is also simple.

2.1. Working of SOMA

The working of algorithm SOMA can be divided into three steps:

1. Parameters are defined.
2. The population is initialized randomly distributed over the search space.
3. The solution space is explored as follows:

At each generation or migration loop, all individuals from population are evaluated using the objective function. The individual with highest fitness value is known as leader and the individual with worst fitness value is known as active. This algorithm moves in migration loops and in each migration loop active individual

Table 1
Best, worst and mean of objective function value, success rate, average time and average number of function evaluations for 100 dimensions.

Prob	Best, worst and mean of objective function value		Success rate	Average time taken in one run (in s)	Average number of function evaluations
Ackley	Best	0.000151	50	0.032	775
	Worst	0.000978			
	Mean	0.000686			
Cosine Mixture	Best	0.000066	50	0.045	550
	Worst	0.000990			
	Mean	0.000566			
Griewank	Best	0.000104	49	0.46	735
	Worst	0.000991			
	Mean	0.000558			
Rasrigin	Best	0.000077	50	0.056	981
	Worst	0.000993			
	Mean	0.000635			
Zakhrov	Best	0.000016	50	0.017	471
	Worst	0.000967			
	Mean	0.000442			
De Jong's function with noise	Best	0.00269	50	0.032	1062
	Worst	0.00968			
	Mean	0.00654			
Sphere	Best	0.000392	50	0.019	638
	Worst	0.000984			
	Mean	0.000507			
Axis parallel hyper ellipsoid	Best	0.000034	50	0.023	756
	Worst	0.000998			
	Mean	0.000516			
Shifted Sphere	Best	−450	49	0.061	760
	Worst	−449.999			
	Mean	−450			
Shifted Rastrigin	Best	−330	50	0.034	587
	Worst	−329.999			
	Mean	−330			
Shifted Griewank	Best	−180	50	0.048	582
	Worst	−179.999			
	Mean	−180			
Shifted Rosenbrock	Best	488.978	0	0.875	5000
	Worst	488.723			
	Mean	488.93			