ISSN:0975-9646

Comparative Analysis of Backtrack Search Optimization Algorithm (BSA) with other Evolutionary Algorithms for Global Continuous Optimization.

Shovan Mandal[#1], Rohit Kumar Sinha[#2], Kamal Mittal[#3]

[#1]*Computer Science and Engineering, #[2]Electrical and Electronics Engineering, #[3]Electrical and Electronics Engineering*
National Institute of Technology, Warangal
Warangal, Telangana, India

Abstract- **In the real world scenario we come across the problem of optimization a number of times. Finding the best solution among the available set of solutions becomes mandatory. A number of numerical techniques are already present in literature which aims at optimizing the result however, they are not feasible to be used in each type of problem. Hence we are tending towards evolutionary algorithms which are more powerful tools to fetch the best results without using any set formulae. A Number of algorithms are already available in literature however they have a problem of getting stuck in local minima or their time of convergence is too high. In this paper I have implemented Backtracking Search Optimization Algorithm (BSA). BSA uses two set of populations i.e. old and new which prevents it from getting stuck into local minima. Its selection, crossover and mutation processes are different from the other methods and it yields the most optimized solution in lesser time. The claim is supported by the results of its comparison with different techniques and BSA is proved to give better results and in lesser time.**

***Keywords*— Backtrack Search Algorithm (BSA), global continuous optimization, Hybrid Ant colony-Genetic Algorithm (GAAPI)**

1. INTRODUCTION

Decision science and the analysis of physical system attach great importance to optimization techniques. GLOBAL optimization in operations research and computer science refers to the procedure of finding approximate solutions, which are considered the best possible solutions, to objective functions [3], subject to constraints on their variables.

Ideally, the approximation is optimal up to a small constant error, for which the solution is considered to be satisfactory. In general, there can be solutions that are locally optimal, but not globally optimal; this situation appears more frequently when the dimension of the problem is high and when the function has many local optima [4]. Consequently, global optimization problems are typically quite difficult to be solved exactly, particularly in the context of nonlinear problems or combinatorial problems. Global optimization problems fall within the broader class of nonlinear programming. It should be noted that approximation algorithms are increasingly being used for problems where exact polynomial algorithms are known but are computationally expensive due to the dimensionality of these problems. This paper focuses on the general global optimization problems in the continuous domain, having a nonlinear objective function that is either unconstrained or that has simple bound constraints.

In the last three decades, a significant research effort was focused on the development of effective and efficient stochastic methods that could reach the nearest global optimal solution. In this class of methods, evolutionary computation (EC) is one of the favourite methodologies, using techniques that exploit a set of potential solutions (called a population) to detect the optimal solution through cooperation and competition among the individuals of the population.

These techniques often find the optima for difficult optimization problems faster than traditional adaptive stochastic search algorithms. The most frequently used population-based EC methods include evolutionary strategies genetic algorithms (GAs) [7]-[8], ant colony optimization (ACO/API)[9]-[10], and particle swarm optimization (PSO) [6]. One of the issues that probabilistic optimization algorithms might face in solving global, highly non convex optimization problems is premature convergence. When the objective function for an optimization problem is non-linear and non-differentiable, evolutionary algorithm (EA) techniques are typically used to find the global optimum. The most commonly used EA optimization techniques are based on swarm intelligence and genetic evolution.

EAs are population-based stochastic search mechanisms that search for near-optimal solutions to a problem. An EA tries to evolve an individual into one with a better fitness value through a 'trial individual'. To generate a trial individual, the EA chooses existing individuals as raw genetic material and combines these using various genetic operators. If the trial individual has a better fitness value than the original individual, the trial individual replaces it in the next-generation population. EAs radically differ from one another based on their strategies for generating trial individuals. Because these strategies have a considerable effect on their problem-solving success and speed, on-going efforts are aimed at developing EAs with faster and more successful problem-solving processes.

The algorithm implemented in this paper, BSA, is a new Evolutionary Algorithm. BSA is a new nature-inspired algorithm is effective, fast and capable of solving different

numerical optimization problems with a simple structure [5]. BSA's unique mechanism for generating a trial individual enables it to solve numerical optimization problems successfully and rapidly. BSA uses three basic genetic operators– selection, mutation and crossover – to generate trial individuals. BSA has a random mutation strategy that uses only one direction individual for each target individual, in contrast with many other genetic algorithms. BSA randomly chooses the direction individual from individuals of a randomly chosen previous generation. BSA uses a non-uniform crossover strategy that is more complex than the crossover strategies used in many genetic algorithms.

This paper uses test set to examine the success of BSA and the comparison algorithms in solving numerical optimization problems. The test set includes 10 widely used standard benchmark problems [2].

2. BSA- BACKTRACK SEARCH OPTIMIZATION ALGORITHM

BSA is a population-based iterative EA designed to be a global minimizer [1]. BSA can be explained by dividing its functions into five processes as is done in other EAs: initialization, selection-I, mutation, crossover and selection-II.

Algorithm 1. General Structure of BSA

```
1. Initialization
repeat
    2. Selection-I
    Generation of Trial-Population
        3. Mutation
        4. Crossover
    end
    5. Selection-II
until stopping conditions are met;
```

2.1 Initialization

BSA initializes the population P with Eq. (1):

$$P_{i,j} \sim U(low_j, up_j) \quad (1)$$

for i = 1,2,3,...,N and j = 1,2,3,...,D, where N and D are the population size and the problem dimension, respectively, U is the uniform distribution and each Pi is a target individual in the population P.

2.2 Selection-I

BSA's Selection-I stage determines the historical population oldP to be used for calculating the search direction. The initial historical population is determined using Eq. (2):

$$oldP_{i,j} \sim U(low_j, up_j) \quad (2)$$

BSA has the option of redefining oldP at the beginning of each iteration through the 'if-then' rule in Eq. (3):

$$\text{if } a < b \text{ then } oldP := P \mid a, b \sim U(0, 1), \quad (3)$$

where := is the update operation. Eq. (3) ensures that BSA designates a population belonging to a randomly selected previous generation as the historical population and remembers this historical population until it is changed. Thus, BSA has a memory. After oldP is determined, Eq. (4) is used to randomly change the order of the individuals in oldP:

$$oldP := permuting(oldP) \quad (4)$$

The permuting function used in Eq. (4) is a random shuffling function.

2.3 Mutation

BSA's mutation process generates the initial form of the trial population Mutant using Eq. (5)

$$Mutant = P + F.(oldP - P), \quad (5)$$

In Eq. (5), F controls the amplitude of the search-direction matrix (oldP - P). Because the historical population is used in the calculation of the search-direction matrix, BSA generates a trial population, taking partial advantage of its experiences from previous generations. This paper uses the value F = 3.rndn, where rndn ~N(0, 1) (N is the standard normal distribution).

2.4 Crossover

BSA's crossover process generates the final form of the trial population *T*. The initial value of the trial population is *Mutant*, as set in the mutation process. Trial individuals with better fitness values for the optimization problem are used to evolve the target population individuals. BSA's crossover process has two steps. The first step calculates a binary integer-valued matrix *(map)* of size *N*. D that indicates the individuals of *T* to be manipulated by using the relevant individuals of *P*. If $map_{n,m} = 1$, where $n \in \{1,2,3\ldots,N\}$ and $m \in \{1,2,3\ldots,D\}$, *T* is updated with $T_{n,m} := P_{n,m}$

In Algorithm-2 (on line 3) indicates the ceiling function, defined as *rnd* ~ *U* (0, 1). BSA's crossover strategy is quite different from the crossover strategies used in EA's and its variants.

Algorithm 2 shows a BSA's unique crossover strategy

Algorithm 2. Crossover Strategy of BSA

```
   Input: Mutant,mixrate,N and D.
   Output: T:Trial-Population.
0  map(1:N,1:D)=1    // Initial-map is an N-by-D matrix of ones.
1  if a < b | a, b ~ U(0, 1) then
2      for i from 1 to N do
3          map_{i,u(1:⌈mixrate·rnd·D⌉)} = 0 | u = permuting((1, 2, 3, ..., D))
4      end
5  else
6      for i from 1 to N do, map_{i,randi(D)} = 0, end
7  end
8  T := Mutant    // Initial T
9  for i from 1 to N do
10     for j from 1 to D do
11         if map_{i,j} = 1 then T_{i,j} := P_{i,j}
12     end
13 end
```

The mix rate parameter (mixrate) in BSA's crossover process controls the number of elements of individuals that will mutate in a trial by using ceil (mixrate. rnd. D) (Algorithm-2, line-3). The function of the mix rate is quite different from the crossover rate used in EA's.

Two predefined strategies are randomly used to define BSA's map. The first strategy uses mixrate (Algorithm 2, lines 2-4). The second strategy allows only one randomly chosen individual to mutate in each trial (Algorithm-2, line-6). BSA's crossover process is more complex than the process used in EA's.

Some individuals of the trial population obtained at the end of BSA's crossover process can overflow the allowed search space limits as a result of BSA's mutation strategy. The individuals beyond the search-space limits are regenerated using Algorithm-3.

Algorithm 3. Boundary Control Mechanism of BSA

```
Input: T, Search space limits (i.e., low_j, up_j)
Output: T
for i from 1 to N do
    for j from 1 to D do
        if (T_{i,j} < low_j) or (T_{i,j} > up_j) then
            T_{i,j} = rnd · (up_j − low_j) + low_j
        end
    end
end
```

2.5 Selection-II

In BSA's Selection-II stage, the T_i's that have better fitness values than the corresponding P_i's are used to update the P_i's based on a greedy selection. If the best individual of P (P_{best}) has a better fitness value than the global minimum value obtained so far by BSA, the global minimizer is updated to be P_{best} , and the global minimum value is updated to be the fitness value of P_{best} . The structure of BSA is quite simple; thus it is easily adapted to different numerical optimization problems.

3. Parameter Settings And Test Functions

In this section of the paper, the performance of the proposed algorithm is investigated, considering a set of 10 benchmark test functions. These test functions are widely used in the scientific literature to test optimization algorithms. Note that most of the test functions have many local minima so that they are challenging enough for performance evaluation.

3.1 Test Functions

Ten widely used functions have been chosen from [2] as test functions, and the proposed algorithm in this paper was tested for all of them. These functions are shown in the Appendix of the paper. A few descriptive characteristics of a class of some very popular test functions (out of the 10 functions) are provided in Table 1. The basic parameters of all 10 test functions are listed in Table 1, including search space limits, their dimension, and their global minimum.

For all 20 test functions, the results obtained by GAAPI are compared to other well-known evolutionary based optimization methods.

3.2 Parameter values for BSA

The values of the parameters of BSA that have been used for the global optimization of the 10 test functions are given. The population size of BSA is variable and depends on the current iteration and the number of unsuccessful sites memorized until the recruitment process.

TABLE I CHARECTERISTICS OF BENCHMARK FUNCTIONS

Test Function	Search Space	Global Minima	Dimension(n)
F1	$[-500,500]^n$	-12569.5	30
F2	$[-5.12,5.12]^n$	0	30
F3	$[-32,32]^n$	0	30
F4	$[-600,600]^n$	0	30
F5	$[-100,100]^n$	0	30
F6	$[-5,5]^n$	-1.0316	2
F7	[-5,10]x[0,15]	0.398	2
F8	$[-2,2]^n$	3	2
F9	$[-5,5]^n$	-78.3324	100
F10	$[-10,10]^n$	0	30

TABLE II PERFORMANCE OF BSA OVER THE 10 TEST FUNCTIONS

Function	Algorithm used and CPU time (s)				
F1	HTGA	CPSO-H6	LEA	GAAPI	BSA
	689.30	658.70	656.30	30.59	9.01
F2	HTGA	CPSO-H6	LEA	GAAPI	BSA
	607.50	557.70	557.20	27.07	8.96
F3	ALEP	CPSO-H6	LEA	GAAPI	BSA
	359.30	326.80	326.10	18.26	9.85
F4	HTGA	CPSO-H6	LEA	GAAPI	BSA
	373.80	368.10	365.60	37.28	9.77
F5	HTGA	CPSO-H6	LEA	GAAPI	BSA
	312.50	242.60	240.20	35.64	9.10
F6	HTGA	ALEP	LEA	GAAPI	BSA
	31.60	31.10	30.80	23.83	6.86
F7	HTGA	ALEP	LEA	GAAPI	BSA
	31.10	31.10	30.60	27.87	6.12
F8	HTGA	ALEP	LEA	GAAPI	BSA
	35.40	34.00	33.50	27.20	6.26
F9	ALEP	CPSO-H6	LEA	GAAPI	BSA
	782.70	685.80	612.30	37.93	29.60
F10	HTGA	CPSO-H6	LEA	GAAPI	BSA
	322.60	243.00	240.80	37.56	9.02

4. Empirical Proof Of Convergence: Results And Analysis

The algorithm was executed in 50 independent runs for each test function, to keep the same base of comparison. The algorithm was implemented in MATLAB R2012a on a Intel CORE^{TM} i5 personal computer with a 3.6 GHz processor. The following data are recorded: the global minimum and the average CPU time of 50 independent runs denoted by CPU. The last analysis component gives a fair indication about the effectiveness of the algorithm in real problems. The aforementioned parameters are generally accepted indicators of performance when referring to heuristic global optimization algorithms. Note that CPU time, together with the PC platform on which the algorithm was executed, is only provided for comparison reasons to other works which used this indicator.

BSA responds very well, particularly for complex functions with higher dimensionality (N=100 or N=30, such as in F1-F5, F9, F10). Table II provides a comparison of the computational time required for BSA and other heuristic methods for determining the global optimal solution. Results on other methods are obtained from [11].

The initials of the algorithms referenced in this paper are presented in Table III. A brief description of some of these

algorithms is presented in [2]. It should be noted that in the literature selected for comparison for the purposes of this work, the same number of function evaluations for each algorithm was not available. Thus, the comparison below gives this measure only to sustain a quasi-comparison on the speed of convergence of different heuristic algorithms toward a near global solution as denoted by the authors as the best-mean solution over a number of independent runs.

TABLE III NOTATIONS OF THE ALGORITHMS USED FOR COMPARISON

Notation	Description
ALEP	Evolutionary programming with adaptive Levy mutation
CPSO-H6	Hybrid cooperative particle swarm optimization. API- special class of continuous domain ant colony optimization search based on the Monmarche approach[10]
LEA	Level-set evolution and Latin squares algorithm
HTGA	Hybrid Taguchi- genetic algorithm
GAAPI	Hybrid Ant Colony-Genetic Algorithm for global continuous optimization

As the computational effort is very important, particularly to actual problems that need to be solved in real time, BSA may be considered as a useful optimization tool based on the computational time required determining the global optimum.

5. CONCLUSION

In this paper, a new algorithm, called BSA, was proposed to solve global unconstrained continuous optimization problems. BSA's algorithmic structure enables it to benefit from previous generation populations by using solutions it has found in the past for a given problem as it searches for solutions with better fitness values. BSA's bio-inspired philosophy is analogous to the return of a social group of living creatures at random intervals to hunting areas that were previously found fruitful for obtaining nourishment.

The values obtained through BSA algorithm are compared with other evolutionary algorithms to prove that BSA is more efficient in terms of CPU utilization time (as it is faster than most of the comparison algorithms) as well as in determining global optimum solution. It was proven that in most of the cases presented in this paper BSA provided satisfactory or optimum solutions, with very little computational effort. The algorithm is recommended for large, complex problems with a dimensionality greater than 30. The detailed tests discussed in

this paper demonstrates that BSA is statistically successful in solving real-valued numerical optimization problems.

The factors responsible for BSA's greater success relative to the comparison algorithms are as follows:

- BSA's mutation and crossover operators produce very efficient trial populations in each generation.
- BSA's generation strategy for the parameter F, which controls the amplitude of the search direction, can produce both numerically large amplitude values necessary for a global search and the small amplitude values necessary for a local search in a very balanced and efficient manner. This clearly enhances BSA's problem-solving ability.
- The historical population (oldP) that BSA uses for the calculation of the search-direction matrix belongs to a randomly selected previous generation. Thus, the historical populations used in more advanced generations include more efficient individuals relative to the historical populations used in older generations. This facilitates BSA's generation of more efficient trial individuals.
- BSA's crossover strategy has a non-uniform and complex structure that ensures creation of new trial individuals in each generation. This crossover strategy enhances BSA's problem-solving ability.
- BSA's boundary control mechanism is very effective in achieving population diversity, which ensures efficient searches, even in advanced generations.

Other Evolutionary Algorithms may further improve the quality of the solution in difficult global optimization problems, but a difficulty in implementation could appear due to the complicated forms of the operators to be used. There may also be value in concentrating on comparisons of BSA to other evolutionary algorithms which may relate to GA, PSO and other local search mechanisms. This study focused mainly on continuous domain optimization problems, so further work can be addressed to see the applicability of the proposed algorithm to discrete as well as constrained optimization problems.

6. APPENDIX

The functions used for testing the proposed algorithm are provided below. These are taken from [2], [11].

$$F_1 = \sum_{i=1}^{n} -x_i \sin(\sqrt{|x_i|})$$

$$F_2 = \sum_{i=1}^{n} (x_i^2 - 10\cdot\cos(2\pi i) + 10)$$

$$F_3 = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi i)\right) + 20 + \exp(1)$$

$$F_4 = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$$

$$F_5 = \sum_{i=1}^{n} x_i^2$$

$$F_6 = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$$

$$F_7 = \left(x_2 - \frac{5.1}{4\pi^2}x_2^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos x_1 + 10$$

$$F_8 = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)]\ X\ [30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 42x_2 - 36x_1x_2 + 27x_2^2)]$$

$$F_9 = \frac{1}{n}\sum_{i=1}^{n}(x_i^4 - 16x_i^2 + 5x_i)$$

$$F_{10} = \sum_{i=1}^{n}|x_i| + \prod_{i=1}^{n}|x_i|$$

REFERENCES

[1] Pinar Civicioglu, "Backtracking Search Optimization Algorithm for numerical optimization problems", Elsevier Trans. Applied Mathematics and Computation pp. 8121- 44, 2013.

[2] Y. Wang and C. Dang, "An evolutionary algorithm for global optimization based on level-set evolution and Latin squares," IEEE Trans. Evol. Comput., vol. 11, no. 5, pp. 579–595, Oct. 2007.

[3] Y.-W. Leung and Y. Wang, "An orthogonal genetic algorithm with quantization for global numerical optimization," IEEE Trans. Evol. Comput., vol. 5, no. 1, pp. 41–53, Feb. 2001

[4] T. Weise, Global Optimization Algorithms: Theory and Applications 2009.

[5] D. Karaboga, B. Basturk, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm, J. Global. Optim. 39 (2007) 459–471.

[6] K. E. Parsopoulos and M. N. Vrahatis, "On the computation of all global minimizers through particle swarm optimization," IEEE Trans. Evol. Comput., vol. 8, no. 3, pp. 211–224, Jun. 2004.

[7] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, 2nd ed. New York: Springer-Verlag, 1994.

[8] Z. Tu and Y. Lu, "A robust stochastic genetic algorithm (StGA) for global numerical optimization," IEEE Trans. Evol. Comput., vol. 8, no. 5, pp. 456–470, Oct. 2004.

[9] M. Dorigo and T. Stützle, Ant Colony Optimization. Scituate, MA: Bradford Company, 2004.

[10] N. Monmarche, G. Venturini, and M. Slimane, "On how Pachycondyla apicalis ants suggest a new search algorithm," Future Gener. Comput. Syst., vol. 16, no. 8, pp. 937–946, Jun. 2000.

[11] I. Ciornei and E. Kyriakides, "Hybrid Ant Colony-Genetic Algorithm(GAAPI) for global Continuous Optimization", IEEE Trans. Systems, MAN, and cybernetics, vol.42, no.1, Feb 2012.