Generation of Test Oracle Using Meta Heuristic Search Technique

Krishna Vir Singh
Department of Computer Science & Engineering
IIMT College of Engineering, Gr. Noida, Uttar Pradesh, India

Dr. Deepak Kumar Singh
Department of Computer Science & Engineering
Sachdeva Institute of Technology, Mathura, Uttar Pradesh, India

Dr. Brij Mohan Singh
Department of Computer Science & Engineering
College of Engineering, Roorkee, Uttrakhand, India

Abstract- It is extremely important that the software to be delivered to the user should be reliable and fault free. This ranges software testing as one of the most important, challenging and dominating activity in the software development life cycle and hence consequent improvement in its usefulness, both with respect to the time and resources, is taken as a key factor by many researchers. An imperative measure of the software testing process is running and evaluating test scenarios. The aim of this part is to assess how well the software under test conforms to its specifications. One of the conducts to achieve this is to generate the test cases and make use of the test oracle to regulate whether a given test case exposes a fault. This procedure incorporates a lot of time. It is very difficult to develop correct, good and unique test cases manually. Use of an automated oracle can achieve the reduction in software testing time and hence the reduction of the cost of the testing process. The use of the Meta Heuristic algorithms has been proposed as a tool for software testing. Optimization is a ubiquitous and spontaneous process which is an integral part of our daily life. Meta Heuristic Algorithms are widely used to tackle black box global optimization problems when no prior knowledge is available. Differential Evolution (DE) is a well-known evolutionary algorithm for solving global optimization problems. Its performance is quite dependent on setting of its control parameters. DE has mainly three control parameters namely, mutation factor (F), crossover rate (CR) and population size (NP). Mutation operator plays a very important role in any evolutionary algorithm. This operator helps to increase search diversity and avoids stagnation in local optima. The main contribution of this thesis report is automatic test oracle design using Neural Network. It uses an extension of differential evolution, called as Modified Differential Evolution (MDE) for test data generation. The performance of Differential Evolution has been improved by introducing a new mutation operator in MDE so that its convergent rate and robustness can be improved. This MDE is applied for solving Travelling Salesman Problem (TSP). The results obtained are fed as an input to train the neural network and then this trained neural network behaves as an automated test oracle. Experimental results show that proposed (MDE) is better and comparable to the other variants of differential evolution. The performance of MDE is analyzed on 30 benchmark functions for four different dimensions viz. 10, 30, 50,100 described in Congress on Evolutionary Computing -2014.

Keywords – Software Testing, Test case automation, Test oracles, Evolutionary Algorithms, Neural Network, Modified Differential Evolution Algorithm, Optimization, Travelling Salesman Problem.

I. INTRODUCTION

Software Testing is a costly and time consuming activity in software development process. Testing involves examining the behavior of a system in order to realize latent faults. Determining the desired correct behavior for a given input is called the oracle problem. For the excellence of the test, the scheming of test cases is important. Test data are generated and testers manually add test oracles and provide a common scenario in software testing. Although completely automatic testing is undecidable problem, conditional or partial automatic testing is quite feasible. Many testing activities can be automated, test cases generation, oracle augmentation, verification, etc.

After an inclusive study made on the existing literature, a lot of limitations/gaps have been found in the area of Software Testing:

- Mainstream works reported for software testing problems have been dealt with statement testing, path testing, branch testing and data flow testing which have their own limitations. Hence an additional attention is required towards a new approach of testing.

- Hybridization of local search and heuristic techniques has been done in most of work. There is restricted effort towards hybridization of META heuristic algorithms in software testing. Hence more prominence is required towards this issue.

- Optimized use of various soft computing techniques to reduce time and cost of software testing process is a promising area of research. Development of heuristic and Meta heuristic are still the major issues related to software testing to automate the testing process to a greater extent.

The work presents the use of an artificial neural network as an automated oracle for Travelling Salesman Problem. A neural network is trained by the back propagation algorithm on a set of test cases applied to the original version of the system. The network training is based on the "black-box" approach, since only inputs and outputs of the system are presented to the algorithm. The trained network can be used as an artificial oracle for evaluating the correctness of the output produced by new and possibly faulty versions of the program.

The test data for neural network were generated using Modified Differential Evolution (MDE), an extension of differential evolution algorithm, on Travelling Salesman problem. MDE improves optimization performance by implementing mutation strategy "DE/current-to-p_{best}" with optional external archive and updating control parameters in an adaptive manner. The parameter adaptation automatically updates the control parameters to appropriate values and avoids a user's prior knowledge of the relationship between the parameter settings and the characteristics of optimization problems. It is thus helpful to improve the robustness of the algorithm.

II. Previous Work

Although Travelling Salesman Problem(TSP) is an old problem, but due to its nature of classic NP-complete combinatorial problem, it intrigues the researchers even today. Due to the vast literature behind the attempts to solve TSP—specific application-orientated approach or generalized approach, it is a cumbersome job to summarize all of those research in short span. Hence, attempt is made here just to have a glimpse of the trends in solving Tsp via time-period. This attempt may reflect only a subset of those research work, but for a brief introduction to the attempts at solving TSP- a slight overview of the trends may be satisfactory.

M. Bellmore and G. L. Nemhauser[3] summarized major trends of solving TSP till the first half century of the 20th century. They have classified the attempts under following categories of algorithmic approach according to solution generation approach:

A. Tour-to-Tour Improvement:

The starting point is an arbitrary tour, say to= (1, 2, n, 1). The solution generation scheme is a rule for finding a better tour that is a 'neighbor' of the present tour. For example, t 1 is the best of the tours that can be generated by interchanging any element 1=2,..., n with 1 . If t_1 = (p, 2,.. n, p) then t_2 is the best tour that can be generated by interchanging i= 1…n, i≠ p, with p. A termination rule could be to stop whenever no improvement can be made. We could then choose any other to and repeat the iterative scheme or apply another, more sophisticated, scheme.

B. Tour Building:

The starting point is an arbitrary node, say i_1. From i, we build a sequence (i_1. i_2. i_k) by successively including other nodes into the sequence. The procedure terminates when a tour is achieved. A very simple scheme of this type is the 'nearest neighbor' rule. From i_1 proceed to the nearest node i_2, from i_2 proceed to the nearest node not yet reached not (i_1 or i_2) from in return to i_l. Clearly, this method of tour building is approximate and there are many variations of the 'nearest neighbor' rule.

C. Subtour Elimination:

The starting point is an optimal solution to the assignment problem under the matrix C. If the solution to the assignment problem is a tour, it is optimal for the travelling salesman problem. If the optimal solution to the assignment problem is not a tour, an iterative scheme is used to eliminate sub tours.

Several algorithms that have produced impressive results. These include:

- Branch-and-bound
- Dynamic programming
- Integer programming
- Gilmore – Gomory method
- Tour to tour approximation

The later research work tried to improve upon these suggested algorithm paths as in [4].
The trend of research in solving TSP focused on nature inspired algorithm during last decade of 20th century and first decade of 21st century. In [5] Zhang evolves that TSP in NP-Hard, recent theoretical research shows a randomized algorithm are designed to the optimal tour in O (n (log n) O (c)) time for every fixed c. The researchers have been either attempting to develop optimization algorithms. Therefore it would seem to be an ideal candidate for nonstandard algorithm approach, such as natural computation. Natural computation is the computational version of the process of extracting ideas from nature to develop computational system. Those that the inspiration from nature for the development of novel problem solving techniques. These natural computational include artificial life, DNA computing, and so on, several solving mechanism via ANN, GA and ACO have been pointed out in the survey paper [5].

Recent survey papers on the advances and current trends in solving TSP by evolutionary methods differential evolution can be seen in [7]. Since, the thesis work deals with solving TSP in part of its proposed work, hence, a brief collection of works to reflect trends in last two decades of so in solving TSP. This is necessary to demonstrate the difficulty of optimization in TSP that generalized satisfactory TSP solution still attracts the researchers. This is not an all inclusive list just a compilation of recent trends as shown in Table 1.

Table 1: Summary of the CEC 14 Test Function

	No	Function	$F_i^* = F_i(x^*)$
Unimodel Function	1	Rotated High Conditioned Elliptic Function	100
	2	Rotated bent Ciger Function	200
	3	Rotated Discuss Function	300
Simple Multunodal Functions	4	Shifted and Rotated Rosenbrock’s Function	400
	5	Shifted and Rotated Acklev's Function	500
	6	Shifted and Rooted Weierstrass Function	600
	7	Shifted and Rotated Gnewank's Function	700
	8	Shifted Rastrwin's Function	800
	9	Shifted and Rotated Rastricrin s Function	900
	10	Shifted Schwefe1's Function	1000
	11	Shifted and Rotated Schwefel-s Function	1100
	12	Shifted and Rotated Katsuura Function	1200
	13	Shitled and Rotated HappyCat Function	1300
	14	Shifted and Rotated HGBat Function	1400
	15	Shifted and Rotated Expanded Griewank's plus Rose.nbrock's Function	1500
	16	Shifted and Rotated Expanded Scaffei's F6 Function Hybrid Function	1600
Hybrid Function	17	Hybrid Function 1 (N=3')	1700
	18	Hvbnd Function 2 (N=3)	1800
	19	Hybrid Function 3 (N=4)	1900
	20	Hybrid Function 4 (N=4)	2000

	21	Hybrid Function 5 (N=5)	2100
	22	Hybrid Function 6 (N=5	2200
Composition Function	23	Composition Function 1(N=5)	2300
	24	Composition Function 2(N=3)	2400
	25	Composition Function 3(N=3)	2500
	26	Composition Function 4(N=5)	2600
	27	Composition Function 5(N=5)	2700
	28	Composition Function 6(N=5)	2800
	29	Composition Function 7(N=3)	2900
	30	Composition Function 8(N=3)	3000
Search Range [-100,100]			

III. PROPOSED ALGORITHM

PROPOSED FRAMEWORK

The performance of differential evolution depends quietly on the setting of control parameters such as the mutation factor (F) and the crossover probability (CR) according to both experimental studies and theoretical analysis. Due to this consideration, different adaptive or self adaptive mechanisms have been introduced to dynamically update the control parameters without a user's prior knowledge of the relationship between the parameters setting and the characteristic of optimization problems.

In addition, the parameter adaptation, if well designed, is capable of improving an algorithms convergence performance by dynamically adapting the parameters to characteristic of different fitness landscapes and the convergence rate can be improved if the control parameters are adapted to appropriate values at different evolution stages of a specific problem Other than the best solution in the current population, historical data is another source that can be used to improve the convergence performance.

The proposed framework in differential evolution algorithm works basically on two general principles which can take into account to develop an efficient mutation operator. They are:

- What to mutate?
- How to mutate?

ADAPTIVE MUTAION CONSTANT

In population based search and optimization methods, considerably high diversity is necessary during the early part of the search to utilize the full range of the search space. On the other hand during the later stages, when the algorithm is converging to an optimal solution fine tuning is important to find the global optimum efficiency. Considering these issues, we propose a new strategy for performance improvement of Differential Evolution. Instead of fixing the value of mutation constant F to a particular value, we make it to be linearly adaptive. The tips of the trial vectors are pointed to simple diverse zones of the search space during the early stages of the search. To meet this objective the value the value of the scale factor has been chosen to very linearly with time as the algorithm proceeds. The value of mutation constant F can be used as $F = f(x_i, g) - f(xpbest,g) / f(xpbest,g)$

PROPOSED ALGORITHM

The computational steps of the proposed modified differential evolution (MDE) are as follows:

```
Algorithm
    01 Begin
    02 Set NPP=20*D, $\mu$CR=0.5;
    03 Create a random initial population {$x_{i,0}$|i=1,2,..NP}
    04 Find the $g_{best}$ =min(Fitness of population)
    05 For g= 1 to G
    06  P=0.05, NP =100;
    07          For i-1 to NP
    08          Generate Cri= randni ($\mu$CR, o.1)
    09         Randomly Choose $x^p_{best,g}$ as one of the 100p% best
                  vectors
    10          Randomly choose xr1, g_=xi,g from current population R
    11          Randomly choose xr2, g_=xr1,g_=xi,g from P U A
    12          calculate F as (f(xig)-f(xpbest,g))/f(xpbest,g);
```

```
13            vi,g = xi,g +Fi·(xpbest,g –xi,g)+Fi·(xr1,g -  ~xr2,g)
14            generate jrand= randint(1,D)
15            for j= 1 to D
16                    if j=jrand or rand(0,1)<CRi
17                    uj,I,g = vj,I,g
18                    else
19                            uj,I,g= xj,I,g
20                    end if
21            end for
22    if f(xi,g) ≤ f(ui,g)
23            xi,g+1=xi,g
24    else
25            xi,g+1=ui,g; xi,g →A; CRi →SCR, Fi →SF
26    end if
27    end for
28    Randomly remove solutions from A so that |A|  ≤ NP
29    μCR = (1-c) · μCR+ c· meanA(SCR)
30    Calculate F as (f(xig)- f(xpbest,g))/f(xpbest,g);
31     End for
32     End
```

EFFECT OF FITNESS BASED ADAPTIVE MUTATION

Fitness based mutation helps to tune the balance between the convergence rate and the robustness of the algorithm. Using fitness information in the mutation operator, we bias the solution in the direction of lowest function value. Using this method it is found that the convergence velocity of the Differential Evolution is locally accelerated so that better solutions are obtained within a accepted convergence time. The key sense of this adaptive mutation is that during the early stages. It is important to explore the whole search space otherwise the best global solutions may be skipped and in later stages it is important to adjust the movements of trial solutions finely so that they can explore the interior of a relatively small space in which the suspected global optimum lies. By making the mutation constant to be time varying this can be achieved. This time varying parametric adaptation scheme attempts to achieve a better tradeoff between the explorative and the exploitative tendencies of Differential Evolution.

TSP USING MDE

Now consider a TSP with pop size 200, number of cities 50 and maximum number of iteration are 10000. A sample run of TSP using MDE is shown in figure. The given algorithm is run 100 times and the Best Tour result obtained after these 100 runs with city number and distances between the cities are feed to neural network using NN toolbox to train the neural network. Now the neural network behaves as TEST Oracle and can be tested by providing appropriate set of inputs and outputs. For this algorithm is as follows:

Procedure of solving TSP using MDE

```
1.  Begin
2.  Create an initial set of xy coordinates for 50 cities
3.  Create an initial population of size 200*2
4.  For g = 1 to numIter
5.          Evaluate each population member (Calculate total distance)
6.          Sort the total distance in ascending order
7.          For i=1 to NP
8.                  Randomly choose $x^{p}_{best,\,g}$ as one of the $100_p$ % best vectors.
9.                  Randomly choose xr1, g_ =  xi, g from current population P.
10.                 Randomly choose ~xr2, g =  xr1, g = xi, g from P U A
11.                 Calculate F as ( f(xig) – f(xpbest, g ))/f(xpbest,g);
12.                 Vi,g = xi,g+Fi · (xpbest,g –xi,g)+Fi · (xr1,g- ~xr2)
13.                 Add the mutated childs received from the mutation in the initial
                    population by replacing the weaker ones.
14.         End
```

15. End
16. Best Tour

IV.RESULT

In this discussion, Modified Differential Evolution algorithm is applied to minimize a set of 30 congress on Evolutionary Computing (CEC) 2014 benchmark functions of dimensions D=10, 30, 50 and 100 as shown in table 3,4,5 and 6. MDE is compared with three adaptive Differential Evolutionary (DE) Algorithms JADE, GaAPADE, L-SHAPE and evolutionary algorithm UMOEAS. Summarized in Table 1 are the 30 benchmark functions. All these functions have an optimal value f* = 0. Parameter setup values of these functions are depicted in table 2.

In all simulate on, we have set the population size NP to be 100 for the values of D, D=10, 30, 50 and 100, while maximum number of function evaluation are 1000D (Max_FES for 10D =100000; for 30D=300000; for 50D=500000; for 100D =1000000). All results reported in this section are obtained based on 51 independent runs.

Table 2: Parameter Setup values

S.no	Test Function	D	NP	MaxGen
1	Ellipsoidal Function	10,30,50,100	100	100000, 3000, 50000, 1000000
2	Bent Cigar Function	10,30,50,100	100	100000, 3000, 50000, 1000000
3	Discuss Function	10,30,50,100	100	100000, 3000, 50000, 1000000
4	Rosenbrock Function	10,30,50,100	100	100000, 3000, 50000, 1000000
5	Ackley Function	10,30,50,100	100	100000, 3000, 50000, 1000000
6	Weierstrass Function	10,30,50,100	100	100000, 3000, 50000, 1000000
7	Griewank Function	10,30,50,100	100	100000, 3000, 50000, 1000000
8	Rastrigin Function	10,30,50,100	100	100000, 3000, 50000, 1000000
9	Step Rastrigin Function	10,30,50,100	100	100000, 3000, 50000, 1000000
10	Schwefel Function	10,30,50,100	100	100000, 3000, 50000, 1000000
11	Schwefel F-7 Function	10,30,50,100	100	100000, 3000, 50000, 1000000
12	Katsuura Function	10,30,50,100	100	100000, 3000, 50000, 1000000
13	Happycat Function	10,30,50,100	100	100000, 3000, 50000, 1000000
14	Hagbat Function	10,30,50,100	100	100000, 3000, 50000, 1000000
15	Griewank-RosenBrock	10,30,50,100	100	100000, 3000, 50000, 1000000
16	Escaffer6 Function	10,30,50,100	100	100000, 3000, 50000, 1000000
17	Hybrid Function 01	10,30,50,100	100	100000, 3000, 50000, 1000000
18	Hybrid Function 02	10,30,50,100	100	100000, 3000, 50000, 1000000
19	Hybrid Function 03	10,30,50,100	100	100000, 3000, 50000, 1000000
20	Hybrid Function 04	10,30,50,100	100	100000, 3000, 50000, 1000000
21	Hybrid Function 05	10,30,50,100	100	100000, 3000, 50000, 1000000
22	Hybrid Function 06	10,30,50,100	100	100000, 3000, 50000, 1000000
23	Composition Function 1	10,30,50,100	100	100000, 3000, 50000, 1000000
24	Composition Function 2	10,30,50,100	100	100000, 3000, 50000, 1000000
25	Composition Function 3	10,30,50,100	100	100000, 3000, 50000, 1000000
26	Composition Function 4	10,30,50,100	100	100000, 3000, 50000, 1000000
27	Composition Function 5	10,30,50,100	100	100000, 3000, 50000, 1000000
28	Composition Function 6	10,30,50,100	100	100000, 3000, 50000, 1000000
29	Composition Function 7	10,30,50,100	100	100000, 3000, 50000, 1000000
30	Composition Function 8	10,30,50,100	100	100000, 3000, 50000, 1000000

a. Comparison of MDE with other Evolutionary Algorithms

The mean of the result obtained by each algorithm for f1-f30 are summarized in tables 3,4,5 and 6. These statistics are calculated for f1-f30, for 1000 iterations.

Important observations about the convergence rate and reliability of different algorithms can be made from the results presented in Tables IV and V. First, these results suggest that the overall convergence rate of MDE is average for D=10 having 11 best results among all for functions f2, f4, f7, f9, 123, f26, f27, f28, f30. MDE achieves the best performance in the case of high dimensional problems (D =50 and 100). For D=30, MDE performs best for functions f2, f5, f23- f30. For D=50, MDE performs best for functions f 1- f7, f 13, f15, f23-f30. For D=100, MDE performs best for functions best for functions f5, f12, f19, f23-26, f28-30. It can also be concluded for figures 21 and 22 that the MDE performs well for composite functions and for higher dimension problems.

B. Parameter values of MDE

MDE introduces its own parameter F, (calculated by extracting 100p% from the population known as pbest, finding their fitness values subtracting these fitness values from the fitness values of $x_{i,g}$ selected randomly from population of size again 100p% and normalizing obtained values by dividing these values by fitness values of $x^{p}_{best,g}$ which determine the greediness of the mutation strategy. It is believed that these two parameters are problem insensitive according to their roles in MDE, it is thus an advantage over the problem-dependent parameter selection (F and CR) in the classic DE. However, it is still interesting to find a range of these parameters which is appropriate for different problems.

Now this extension is applied to solve travelling salesman problem. The results obtained after a single run of the MDE is shown below:

Table 3: Experimental Results of 10 dimensional problem f1-f30, Averaged over 51 independent runs.

Fun.	**Gen**	**MIDE**	**JADE**	**GaAPADE**	**L-SHAPE**	**UMOEAS**
F1	1000	6.28e+07	4.55e+07	6.7e+07	1.79e+03	8.38e+07
F2	1000	2.05e+09	3.3e+09	4.2e+09	3.19e+09	5.28e+09
F3	1000	1.7e+04	3.35e+04	5.94e+04	4.65e+04	1.75e+04
F4	1000	3.49e+02	3.97e+02	3.8e+02	3.56e+02	6.6e+02
F5	1000	20.57	20.6	20.68	20.6	20.72
F6	1000	10.15	10.10	10.7	10.4	11.18
F7	1000	57.82	59.59	61.57	52.12	85.93
F8	1000	77.13	68.26	78.85	72.19	83.33
F9	1000	63.27	76.73	82.28	80.08	74.88
F10	1000	1.4e+03	1.50e+03	1.80e+03	1.79e+03	1.5e+03
F11	1000	1.7e+03	1.8e+03	1.95e+03	1.79e+03	1.95e+03
F12	1000	1.71	2.0	2.41	2.45	2.45
F13	1000	2.16	2.36	2.45	2.22	2.99
F14	1000	6.89	15.28	16.05	14.24	22.76
F15	1000	1.3e+02	1.5e+03	3.54e+03	1.84e+03	8.76e+03
F16	1000	4.47	3.95	4.03	4.07	4.18
F17	1000	5.91e+05	1.13e+06	1.43e+06	9.14e+05	5.95e+05
F18	1000	8.1e+04	2.89e+06	5.5e+06	4.01e+06	1.55e+06
F19	1000	38.46	13.33	14.05	14.78	20.72
F20	1000	1.7e+04	7.2e+04	2.15e+05	1.37e+05	1.15e+05
F21	1000	1.18e+06	2.2e+05	3.09e+05	1.84e+05	2.99e+05
F22	1000	2.6e+02	2.4e+02	3.04e+02	2.59e+02	3.07e+02
F23	1000	2.08e+02	3.8e+02	4.1e+02	3.93e+02	2.34e+02
F24	1000	2.00e+02	1.9e+02	2.02e+02	1.97e+02	1.98e+02
F25	1000	2.00e+02	2.03e+02	2.06e+02	2.05e+02	1.99e+02
F26	1000	2.00e+02	1.01e+02	4.27e+02	1.01e+02	1.02e+02
F27	1000	2.12e+02	3.26e+02	1.01e+02	4.15e+02	3.6e+02
F28	1000	2.19e+02	1.02e+03	4.7e+02	1.03e+03	3.6e+02
F29	1000	7.1e+06	1.13e+06	5.5e+05	1.36e+06	2.77e+06
F30	1000	1.24e+06	1.31e+04	1.45e+04	1.57e+04	2.8e+04

Table 4: Experimental Results of 30 dimensional problem f1-f30, Averaged over 51 independent runs.

Fun.	**Gen**	**MIDE**	**JADE**	**GaAPADE**	**L-SHAPE**	**UMOEAS**
F1	1000	3.47e+08	7.24e+08	1.08e+09	1.1.e+09	7.18e+08
F2	1000	1.75e+10	4.29e+10	4.87e+10	5.60e+10	4.19e+10
F3	1000	8.24e+04	9.01e+04	1.8e+05	1.75e+06	8.40e+04
F4	1000	2.35e+03	5.81e+03	8.28e+03	1.63e+05	5.87e+03
F5	1000	20.85	21.0	21.06	8.78e+03	21.14
F6	1000	36.92	38.9	40.49	21.13	40.36
F7	1000	1.7e+02	3.78e+02	4.7e+02	42.23	3.74e+02
F8	1000	2.80e+02	3.09e+02	3.07e+02	4.94e+02	3.194e+02
F9	1000	2.8e+02	3.69e+02	3.86e+02	3.53e+02	3.47e+02
F10	1000	6.5e+03	6.57e+03	7.36e+03	7.74e+03	7.2e+03
F11	1000	7.5e+03	7.67e+03	7.87e+03	7.74e+03	8.04e+03
F12	1000	2.015	3.25	3.91	7.74e+03	7.2e+03
F13	1000	3.43	5.30	5.61	6.04	5.05
F14	1000	68.5	1.40e+02	1.58e+02	1.77e+02	1.37e+02
F15	1000	4.5e+03	3.07e+05	1.30e+06	8.06e+05	1.05e+05
F16	1000	13.98	13.47	13.59	13.66	13.67
F17	1000	1.9e+07	3.18e+07	5.78e+07	5.50e+07	3.55e+07
F18	1000	5.9e+08	8.17e+08	1.76e+09	1.63e+09	8.24e+08
F19	1000	1.13e+02	2.25e+02	3.18e+02	3.26e+02	2.58e+02

F20	1000	1.29e+05	1.18e+05	4.9e+05	3.35e+05	1.61e+05
F21	1000	8.9e+06	9.47e+06	1.87e+07	1.85e+07	1.06e+07
F22	1000	1.6e+03	1.29e+03	1.7e+03	1.64e+03	1.39e+03
F23	1000	2.17e+02	5.53e+02	7.86e+02	7.34e+02	2.78e+02
F24	1000	2.02e+02	3.46e+02	3.89e+02	3.80e+02	2.07e+02
F25	1000	2.00e+02	2.58e+02	2.8e+02	2.87e+02	2.01e+02
F26	1000	2.00e+02	1.05e+02	1.05e+02	1.11e+02	1.07e+02
F27	1000	2.29e+02	9.20e+02	1.3e+02	1.25e+03	6.39e+02
F28	1000	2.3e+02	5.02e+03	2.8e+03	5.72e+03	6.39e+02
F29	1000	2.08e+07	1.24e+08	4.3e+07	2.05e+08	8.63e+07
F30	1000	1.39e+06	8.61e+05	1.33e+06	4.07e+02	9.78e+05

Table 5: Experimental Results of 50 dimensional problem f1-f30, Averaged over 51 independent runs.

Fun.	Gen	MIDE	JADE	GaAPADE	L-SHAPE	UMOEAS
F1	1000	3.47e+08	1.18e+09	2.56e+09	2.64e+09	1.24e+09
F2	1000	1.89+10	7.70e+10	1.0e+11	1.35e+11	6.63e+10
F3	1000	1.40e+05	4.15e+05	2.9e+05	2.70e+05	1.255e+05
F4	1000	2.48e+03	1.49e+04	2.83+04	3.49e+04	1.20e+04
F5	1000	20.69	21.1	21.20	21.27	21.26
F6	1000	55.54	67.3	72.01	75.96	70.40
F7	1000	1.57e+02	7.38e+02	1.06e+03	1.30e+03	6.33e+02
F8	1000	4.89e+02	5.42e+02	5.48e+02	6.82e+02	5.67e+02
F9	1000	5.2e+02	6.41e+02	7.37e+02	8.06e+02	6.33e+02
F10	1000	1.18e+04	1.17e+04	1.38e+04	1.42e+04	1.34e+04
F11	1000	1.18e+04	1.35e+04	1.44e+04	1.48e+04	1.44e+04
F12	1000	1.61	3.50	4.75	4.70	3.82
F13	1000	2.05	5.73	6.55	7.79	5.33
F14	1000	38.57	1.95e+02	2.66e+02	3.48e+02	1.22e+02
F15	1000	1.04e+04	1.35e+06	9.19e+06	9.11e+06	1.11e+06
F16	1000	22.74	22.92	23.41	23.35	23.26
F17	1000	7.06e+07	9.91e+07	2.16e+08	2.80e+08	9.93e+07
F18	1000	2.3e+09	2.17e+09	6.6e+09	7.78e+09	1.77e+09
F19	1000	3.37e+02	4.45e+02	9.6e+02	1.06e+03	4.37e+02
F20	1000	1.1e+05	1.34e+05	9.06e+05	7.10e+05	1.91e+05
F21	1000	1.25e+07	2.10e+07	7.27e+07	6.96e+07	2.76e+06
F22	1000	3.25e+03	2.74e+03	4.14e+03	5.34e+03	2.79e+03
F23	1000	2.15e+02	8.16e+02	1.5e+03	1.54e+03	2.89e+02
F24	1000	2.04e+02	4.64e+02	5.9e+02	5.96e+02	2.14e+02
F25	1000	2.00e+02	3.29e+02	1.13e+02	4.34e+02	2.02e+02
F26	1000	2.00e+023	1.22e+02	2.3e+03	3.09e+02	1.29e+02
F27	1000	2.2e+02	2.13e+03	6.7e+03	2.40e+03	6.98e+02
F28	1000	2.3e+07	9.69e+03	6.58e+02	1.23e+04	9.02e+02
F29	1000	2.08e+07	4.17e+08	252650000	1.03e+09	1.71e+08
F30	1000	1.24e+06	4.17e+06	6.06e+06	1.27e+07	4.28e+06

Table 6: Experimental Results of 100 dimensional problem f1-f30, Averaged over 51 independent runs.

Fun.	Gen	MIDE	JADE	GaAPADE	L-SHAPE	UMOEAS
F1	1000	6.59e+09	3.66e+09	0	8.35e+09	1.8e+09
F2	1000	7.35e+09	2.13e+11	0	3.57e+11	1.11e+11
F3	1000	7.5e+07	4.15e+05	0	5.88e+05	3.01e+05
F4	1000	1.29e+05	4.32e+04	0	1.00e+05	1.68e+04
F5	1000	21.42	21.3	0	21.40	21.39
F6	1000	1.5e+03	1.49e+02	0	1.63e+02	1.5e+02
F7	1000	3.35e+03	1.94e+03	0	3.23e+03	1.03e+03
F8	1000	1.59e+03	1.27e+03	0	1.56e+03	1.14e+03
F9	1000	9.77e+02	1.48e+03	0	4.07e+02	1.24e+03
F10	1000	3.2e+04	2.70e+04	0	3.15e+04	2.87e+04
F11	1000	3.18e+04	3.06e+04	0	3.24e+04	2.8e+04
F12	1000	7.76	4.15	0	5.06	4.24
F13	1000	10.01	7.42	0	9.94	5.46
F14	1000	5.4e+02	4.37e+02	0	3.86e+02	2.36e+02
F15	1000	1.1e+03	1.53e+03	0	6.52e+07	1.4e+06

F16	1000	46.37	47.1	0	47.62	47.59
F17	1000	1.8e+08	39437153	0	1.08e+09	2.2e+08
F18	1000	2.26e+09	9.93e+09	0	3.07e+10	2.7e+09
F19	1000	5.15e+02	1.88e+03	0	5.45e+03	9.2e+02
F20	1000	3.89e+05	1.02e+06	0	3.38e+06	5.07e+05
F21	1000	3.9e+07	1.63e=08	0	4.62e+08	9.07e+07
F22	1000	73.10	6.96e+03	0	3.49e+04	5.7e+03
F23	1000	2.08e+02	1.62e+03	0	3.54e+03	2.7e+02
F24	1000	2.06e+02	8.73e+02	0	1.20e+03	2.24e+02
F25	1000	2.00e+02	6.21e+02	0	9.68e+02	2.02e+02
F26	1000	2.00e+02	4.51e+02	0	7.77e+02	1.98e+02
F27	1000	1.18e+03	3.26e+02	0	4.98e+03	3.63e+02
F28	1000	1.36e+03	2.42e+04	0	3.02e+06	7.68e+02
F29	1000	3.5e+08	1.76e+06	0	1.36e+06	1.96e+08
F30	1000	5.99e+07	3.02e+07	0	1.11e+08	1.7e+07

V. CONCLUSION

In this paper we have developed an extension of differential evolution algorithm, called as Modified Differential Evolution (MDE). We have tested the performance of the proposed algorithm for 30 benchmark functions discussed in Congress on Evolutionary Computing (CEC) -2014 for 10, 30, 50, 100 dimensions.

The algorithm works comparably for the three versions of differential evolution L-SHADE, JADE and GaAPADE and an evolutionary algorithms viz. UMOEA for 10 dimensions, but it works better than these algorithms for dimensions 30, 50 and 100 for most of the functions. This modified algorithm has been applied to solve travelling salesman problem.

The results obtained are feed as an input to the neural network to train the network. Now this trained network behaves as an automated oracle for testing. The neural network is shown to be a promising method of testing software application provided that the training data have a good coverage of the input range. The back propagation method of training the neural network is a relatively rigorous method capable of generalization, and one of its properties ensures that the network can be updated by learning new data.

REFERENCES

[1] L. Baresi, M. Pezz. An Introduction to Software Testing. Electronic Notes in Theoretical Computer Science 148 (2006) 89111

[2] E. Diaz, J. Tuya, R. Blanco. Automated software testing using a meta heuristic technique based on tabu search. Proceedings of 18th IEEE International Conference on Automated Software Engineering, 2003, pp. 310313.

[3] S. R. Schach. Testing: Principles and Practice. ACM Computing Surveys, Vol. 28, No. 1, March 1996.

[4] T. Mantere, J.T. Alander. Evolutionary software engineering, a review. Applied Soft Computing 5 (2005) 315331.

[5] P. McMinn. Search-Based Software Testing: Past, Present and Future. In the Proceedings of 2011 Fourth International Conference on Software Testing, Verification and Validation Workshops.

[6] M. Catelani, L. Ciani , V. L. Scarano, A. Bacioccola. Software automated testing: A solution to maxi-mize the test plan coverage and to increase software reliability and quality in use. Journal of Computer Standards & Interfaces 33 (2011) 152158.

[7] G. Fraser, A. Zeller. Mutation-Driven Generation of Unit Tests and Oracles. IEEE Transactions on Software Engineering (Volume:38 , Issue: 2) March-April 2012.

[8] M. A. Ahmeda, I. Hermadib. GA-based multiple paths test data generator. Computers & Operations Research 35 (2008) 3107 3124

[9] J.A. Whittaker. What is software testing? And why is it so hard?. IEEE Software 17 (2000) 7079

[10] S. R. Shahamiri, W. M. N. W. Kadir, S. Ibrahim, S. Z. M. Hashim. An automated framework for software test oracle. (In press as: S.R. Shahamiri et al., An automated framework for software test oracle, Inform. Softw. Technol. (2011))

[11] L. A. Zadeh. "Fuzzy logic, neural networks, and soft computing". Communications of the ACM, Vol. 37, pp. 77-84, 1994.

[12] T. Weise, M. Zapf, R. Chiong, A. J. Nebro. Nature-Inspired Algorithms for Optimization Series: Studies in Computational Intelligence, Vol. 193 (2009).

[13] I. H. Osman and G. Laporte. Meta heuristics: a bibliography. Annals of Operational Research, Vol. 63 (1996) 513-628. Meta heuristics in Combinatorial Optimization Editors: G. Laporte and LK Osman. Baltzer Science Publications, The Netherlands.

[14] A. K. Dhingra, Multi-objective flow shop scheduling using metaheuristics, Ph. sham, 2011.

[15] J.H. Holland. Genetic Algorithm. Scientific American July 1992.

[16] D. E. Goldberg. Genetic Algorithms in Search, Optimization and Machine Learning Addison-‘$‘Wesley, 1989.

[17] J. Kennedy and R. Eberhart, Particle swarm optimization, IEEE International Conference on Neural Networks, IEEE Press, pp. 19421948, 1995

[18] N. Narmada and D. P. Mohapatra Automatic Test data generation for data flow testing using Particle Swarm Optimization, Communications in Computer and Information Science, Vol. 95, No. 1, pp. 1-12, • 2010.

[19] R. Storn, K. Price. Differential Evolution – A simple and efficient adaptive scheme for global opti-J. Globtd Optimization, vol. 11, no. 4, pp. 341359, 1997.

[20] E. Mezura-Montes, J. Vel azquez-Reyes and C. A. C. Coello. A Comparative Study of Differential Evolution Variants for Global Optimization". ACM proceedings from GECC006, July 812, 2006, Seattle, Washington, USA.

[21] V. Feoktistov and S. lanaqi. Generalization of the Strategies in Differential Evolution. In Proceedings of the 18th International Parallel and Distributed Processing Symposium (1PDPS 2004), 2004, Santa Fe, New Mexico, USA, page 165a, New Mexico, USA, April 2004. IEEE Computer Society.
[22] K. V. Price. An introduction to differential evolution. InD. Come, M. Dorigo, and F. Glover, editors, New Ideas in Optimization, pages 79108. Mc Graw-Hill, UK, 1999.
[23] R. Storm Differential evolution design of an 11R-filter with requirements for magnitude and group de-lay. In Proceedings of 1FFF International Conference on Evolutionary Computation (10EC96), pages 268273 New York, NY, USA, May 1996JEEE.
[24] B. V. Babu and S. A. Munawar. Optimal Design of Shell-and-114*e Heat Exchangers by Different Strategies of Differential Evolution. .11-chnical Report P1LAN1-33.4 031. Department of Chemical Engineering Birla Institute of Technology and Science. Rajasthan. India.. 2001.
[25] A. Qin and P. Suganthan. Self-adaptive Differential Evolution Algorithm for Numerical Optimization. In Proceedings of the Congress on Evolutionary Computation 2005 (CEC2005), volume 1, pages 630636. Piscataway New Jersey. September 2005. Edinburgh, UK., IFRF. Service Center.
[26] J. Vesterstrom and R. Thomsen. A Comparative Study of Differential Evolution Particle Swarm Optimization and Evolutionary Algorithms on Numerical Benchmark Problems. In Proceedings of the IEEE Congress on Evolutionary Computation (CEC2004), volume 3, Pages 19801987, Piscataway, New Jersey, June 2004. Portland Oregon, USA, IFFF. Service Center.
[27] J. Ronkkonen. S. Kukkonen, and K. V. Price. Real-Parameter Optimization with Differential Evolution. In Proceedings of the Congress on Evolutionary Computation 2005 (CEC2005), volume 1. pages 567574, Piscataway, New Jersey, September 2005. Edinburgh, UK,, IFFE Service Center.
[28] S. Kukkonen and J. Lampinen. GDE3: The third Evolution Step of Generalized Differential Evolution. In Proceedings of the Congress on Evolutionary Computation 2005 (CEC2005), volume 1, pages 239246, Piscataway, New Jersey, September 2005. Edinburgh, UK,, IEEE Service Center.
[29] W. J. Zhang and X.-F. Xie. DEPSO: Hybrid Particle Swarm with Differential Evolution Operator. In Proceedings of the JEFF International Conference on Systems, Man and Cybernetics (SMC2003), volume 4, pages 38163821. Washington DC, USA, !REF., October 2003.
[30] N. Noman and H. Iba. Enhancing Differential Evolution Performance with Local Search for High Dimensional Function Optimization. In H.-G. e. Beyer, editor, Proceedings of the Genetic and Evolutionary Computation Conference (GECCO2005), pages 967974, New York, June 2005. Washington DC, USA, ACM Press.
[31] L. T. Bui, Y. Shan, F. Qi, and H. A. Abbass. Comparing Two Versions of Differential Evolution in Real Parameter Optimization. Technical Report TR-ALAR-200504009, School of Information Technology and Electrical Engineering, University of New South Wales, Northcott Drive, Campbell, Canberra, ACT 2.