A Comparative Study of Bat and Cuckoo Search Algorithm for Regression Test Case Selection

Arvinder Kaur
University School of Information and Communication Technology
GGSIP University
New Delhi, India
arvinderkaurtakkar@yahoo.com

Arun Prakash Agrawal
Amity School of Engineering and Technology
Amity University Uttar Pradesh
Noida, India
apagrawal@amity.edu

Abstract- Enhancing the software by either adding new functionality or deleting some obsolete capability or fixing the errors is called software maintenance. As a result, the software may function improperly or unchanged parts of the software may be adversely affected. Testing carried out to validate that no new errors have been introduced during maintenance activity is called Regression Testing. It is acknowledged to be an expensive activity and may account for around 60 -70% of the total software life cycle cost. Reducing the cost of regression testing is therefore of vital importance and has the caliber to reduce the cost of maintenance also. This paper evaluates the performance of two metaheuristic algorithms- Bat Algorithm and Cuckoo Search Algorithm for selecting test cases. Factors that we have considered for performance evaluation are the number of faults detected and the execution time. The domain of study is the flex object from the Benchmark repository – Software Artifact and Infrastructure Repository. Extensive experiments have been conducted to collect and analyze the results. A Statistical test, F-test has also been conducted to validate the research hypothesis. Results indicate that the Cuckoo Search Algorithms perform a little better than Bat Algorithm.

***Keywords*-Software Maintenance, Regression Test Case Selection, Bat Algorithm, Cuckoo Search Algorithm.**

I. INTRODUCTION

Optimization is required in every realm of life. Most of real world optimization problems are multi-objective. Regression test case selection is one such multi-objective optimization problem that has attracted the researchers since last three decades due to the following reasons:

a. Executing all the Test cases will require an unacceptable amount of time whereas there will be pressure of delivery timelines

b. Many Team Members will have to be aligned for Testing which is again not possible

c. Both the above will lead to significant amount of time every time the Regression Testing is done

Being hard in nature, they require sophisticated optimization tools to tackle with. Meta-heuristics are such stochastic algorithms that can be used to solve such problems. They help us to achieve near optimal solutions in acceptable time limit, but with no guarantee. These algorithms are expected to work most of the time but not all of the time. [28] An alternative approach may be identifying a subset of test cases that provide the same fault coverage and/or statement coverage of the software, according to some criteria, as the original test suite [25]. In order to resolve the above mentioned challenges, the Regression Test Selection technique needs to be optimally designed so as to meet the objectives of Regression Testing.

This paper focuses on the use of two nature inspired evolutionary approaches- bat and cuckoo search algorithms for test suite optimization. The algorithms make use of the test history to generate initial population. Fitness value is calculated using fault coverage and running time of test cases and only fit tests are carried out to the successive generations to reduce the test suite until the stopping criterion is met. We then compared the results obtained from both the algorithms and found that the cuckoo search algorithm performs a little better than the bat algorithm. The domain of enquiry in this paper is the flex object which is a lexical analyzer. It has become a benchmark object to conduct such study and is available from open source Software Artifact Infrastructure repository [3]. Experiments were conducted on five versions of the flex object.

Organization of the Rest of the paper is as follows: Section 2 discusses the related work carried out by previous researchers in various areas. Section 3 briefly discusses the Regression Testing and the Regression Test Case selection Problem. Section 4 and 5 give an overview of the Bat and Cuckoo Search algorithms respectively. Section 6 presents experimental design, experimental setup, and discussion on the results obtained. Threats to validity are discussed in section 7. Conclusion and Future scope in section 8 conclude the paper.

II. RELATED WORK

Nachiyappan et al. in [2] investigated the use of Genetic Algorithm for test suite reduction using coverage and running time of test cases as the fitness criteria.

In [4] Nakamura et al. proposed a nature inspired technique combining the exploration power of bats with optimum path forest classifier and applied the same in the area of feature selection for the first time. The results obtained outperformed other swarm based techniques.

Wang et al. in [5] proposed a hybrid metaheuristic approach to solve the global numerical optimization problems. They improved the basic bat algorithm by including the pitch adjustment operator from harmony search algorithm to speed up the convergence of bat algorithm.

In [6] Khan et al. demonstrated the superiority of bat algorithm in training neural networks over genetic algorithm, particle swarm optimization, back propagation and levenberg-marquardt in the field of supervised learning.

Very recently Fister et al. applied bat algorithm in planning the sports training sessions which has always been a challenging task for coaches [7]. Their method is based on the analysis of the data gathered by sports watches like heart rate etc during the sports activities.

Xie et al. proposed a hybrid bat algorithm for solving multiple runways aircraft landing problem [8]. Proposed algorithm obtained high quality solutions for instances up to 500 aircrafts.

In [9] Ye et al. applied bat algorithm to obtain optimal threshold values for segmentation. They tested the proposed method and compared the results with methods based on fuzzy entropy and optimized by ABC, GA, and PSO.

Recently Srivastava et al. [10] applied bat algorithm for predicting test effort estimation affected by several factors like test team productivity, size of system, testing strategy, and complexity of the system and many others.

In [11] Yildiz et al. applied cuckoo search algorithm for solving manufacturing optimization problem by optimizing machining parameters and demonstrated the effectiveness of cuckoo search.

Durgun et al. in [12] demonstrated the application of cuckoo search in the area of structural design optimization of vehicle components for improving cost and fuel efficiency.

In [13] Layeb et al. proposed a quantum inspired cuckoo search QICSA to solve the knapsack problem and demonstrated the effectiveness of their approach in achieving good quality solutions.

Quaarab et al. in [14] proposed a discrete cuckoo search algorithm and demonstrated its superiority in solving travelling salesman problem.

Burnwal et al. in [15] applied cuckoo search algorithm with a slight modification in levy flight operator to solve a standard flexible manufacturing scheduling problem to minimize the penalty cost caused by delay and to maximize the utilization of machine.

Chandrasekaran et al. in [16] proposed a hybrid approach by integrating cuckoo search algorithm with fuzzy systems to solve the multi-objective unit commitment problem considering factors like fuel cost, emission and level of system reliability and demonstrated the effectiveness of the proposed approach.

A version of cuckoo search algorithm was proposed by Gherboudj et al. in [17] using a sigmoid function to solve the 0-1 knapsack problem.

Marichelvam et al. proposed a hybrid cuckoo search algorithm in [18] for solving flow shop scheduling problem.

Test effort estimation has always been a fascinating area of research. Srivastava et al. in [19] proposed a model for test effort estimation using cuckoo search algorithm.

In [20] Srivastava et al. proposed an approach for generating test data using cuckoo and tabu search algorithms.

Srivastava et al. in [21] proposed an approach to generate optimized test sequences which obtain 100% software coverage using cuckoo search algorithm.

In [22] Nagar et al. applied cuckoo search algorithm for test case selection and prioritization on a random test data set.

It can be easily observed that the literature lacks works applying bat and cuckoo search algorithms into the regression test case selection problem. For software test effort estimation, we can find a very few papers using these two algorithms by Srivastava et al., and for bat algorithm, there is no previous work in the area of regression testing to the best of our knowledge. Hence this is a promising area of study and we intend to explore it further by deploying new strategies to take this into account.

III. REGRESSION TESTING

As the software is modified due to any kind of maintenance activity performed, in order to be confident that the no new errors have been introduced, we need to perform regression testing. The primary objective of regression testing is to assure that modifying one part of the software does not adversely affect the other parts of the software [23].

A. Regression Test Case Selection

Large test suites contain some redundancies as the same requirement/fault may be covered by two or more test cases. Hence it is advisable to reduce the test suite [24]. Since manual test case selection can be time consuming and error prone, we consider it as a search based optimization problem and try to solve it with the help of metaheuristic algorithms [29]. In this paper, we have used fault coverage and running time of test cases as the test adequacy criteria.

The aim is to maximize the Fault coverage. Let T = {TI, T2, …..., Tn} be a test suite and F = {Fl, F2, ….. ,Fm} indicates complete fault set. F(Ti) indicates a function which returns the faults coverage of a test case. Then, the fault coverage of a test suite is given by –

$$\text{Fault Coverage} = 100 * \bigcup_{t=1}^{s} \{F(Ti)\}/k \qquad (1)$$

Where s reflects the number of selected test cases and k the total number of faults.

The execution time should be minimized. Execution time is defined as the total time required to execute a test [3]. The total execution time of the selected test cases is given by

$$\sum_{i=1}^{s} Time_i \qquad (2)$$

Where $Time_i$.is the execution time of ith test case.

However test case selection could reduce the rate of detection of faults because the effectiveness of a test suite could decrease with the decrease in test suite size.

IV. BAT ALGORITHM

Bat Algorithm combines the good features of other nature inspired metaheuristics like Harmony Search, Firefly Algorithm and Simulated Annealing [26]. Bat Metaheuristic Algorithm is based on the echolocation property of micro bats. This property guides the foraging behavior of micro bats and helps them to find their prey and allows them to distinguish between different types of insects even if it is too dark [27].

A. Echolocation Property of Bats

Most of the microbats eat insects [26]. They make use of the echolocation to communicate with each other, recognize type of insects, estimate the distance to the prey, and to avoid obstacles in the dark [27]. Bats listen for the echo bounced back from the surrounding objects of the loud sound pulses emitted by them [26]. These pulses vary in frequency and help them decide their hunting strategy [26]. These are very short pulses. The loudness is high while searching for the prey and decreases as they home towards the prey [26].

Microbats build a three dimensional scenario of their surrounding by using the time difference between the emission of the pulse and receiving of the echo. With this they can approximate the distance, orientation, type and also the moving speed of their target insect. For simplicity, following assumptions have been made:

1. Echolocation property is used by all the bats to calculate distance to the prey, and can differentiate between food/prey and the background obstacles.
2. Bats fly in random directions to search for their prey with initial velocity, position, frequency, and wavelength and pulse loudness. Wavelength or frequency and pulse emission rate r are automatically adjusted according to the proximity of the target.
3. Loudness of emitted pulses vary from a large positive value A_0 to a minimum value A_{min}.

Algorithm 1. Pseudo code of the bat algorithm

```
1: Initialize bat position $x_i$ and velocity $v_i$
2: Decide frequency fi
3: Initialize pulse emission rate r and loudness A
4: repeat
5:      Generate new solutions by adjusting frequency and
        updating velocity and location
6:      if rand > $r_i$ then
7:          Select a solution from best solutions
8:          Generate new local solution around selected
            best  solution
9:      end
10:     Generate new solution by flying randomly
11:     if (rand < Ai and f(xi) < f(x*)) then
12:         Accept the new solutions
13:         Decrease Ai, increase $r_i$,
14:     end
15:     Rank the bats and find the current best x*
16: until termination criteria is met;
17: Postprocess results and visualization
```

V. CUCKOO SEARCH ALGORITHM

Cuckoo Search algorithm also belongs to the Nature inspired optimization metaheuristics. It was introduced by Young and Deb in 2009 and has proven to be very promising for solving many hard real world optimization problems [29]. Aggressive reproduction strategy is the main attraction of the cuckoos. They believe in parasitic breeding. It is a kind of parasitism in which a cuckoo lays its eggs in the nest of host species. Some cuckoo species lay their eggs in other bird's nests and may remove the host bird's eggs to increase the hatching probability of their own eggs [29]. Some host birds do not like intruders and conflict with them. In this case either the host bird will throw their eggs out or may simply abandon its nest and build a new nest at some other place.

A. Description of the Algorithm

It is a population based global search stochastic metaheuristic. Each egg corresponds to a potential solution in cuckoo search algorithm. Natural systems are complex and simplification of natural systems is required to successfully implement them by computer algorithms. We make the following three assumptions to simplify the cuckoo search algorithm:

1. Each cuckoo lays only one egg at a time and can leave this egg in a randomly chosen nest.
2. In order to maintain the elitist property, the best nest with the highest quality of eggs (solutions) will make it to the next generations.
3. The number of available host nests to lay eggs is fixed. Cuckoo's egg can be identified by host bird with a probability p_a Є [0, 1]. If identified, the host bird can either throw it away or the host may abandon its own nest and builds a new nest at some other location.

Based on the three assumptions, the basic steps of the cuckoo search can be summarized as below:

Algorithm 2. Pseudo code of the cuckoo search algorithm

```
1: Start
2: Objective function f(x), x = $(x_1,x_2,……,x_u)^T$;
3: Generate initial population of n host nests $x_i$ (i=1,2,…….,n);
4: While (t < Maximum no. of generations) or (stopping criteria is
   met);
5:    Randomly select one cuckoo via Lévy flights
6:    Evaluate its fitness Fi
7:    Choose a nest among n (say j) available nests randomly
8:    If (Fi  > Fj)
9:        Replace j by the new solution;
10:   end
11: Fraction $p_d$ of worse nests are abandoned and new nests are
    being built;
12:   Keep the best solutions or nests with quality solutions;
13:   Rank the solutions and find the current best
14: end while
15: Post process and visualize results
16: End
```

VI. EXPERIMENTAL DESIGN AND RESULTS

This section discusses the experiment conducted to evaluate the bat and cuckoo search algorithms performance. The context of our study consists of five versions of the Flex- Fast Lexical Analyzer program available from the software artifact infrastructure repository as the subject program. The versions used are numbered from v1 to v5.

A. Experimental Design

We have formulated the following two research questions:

RQ1: Is there any difference in the performance of both the algorithms for the regression test case selection problem?

RQ2: How much effort can we save in terms of time through regression test case selection?

In order to answer our research question 1, we have formulated the following null and alternate hypothesis:

Ho: There is no significant difference in the performance of bat and cuckoo search algorithm.

Ha: One algorithm performs significantly different than other.

B. Experimental Setup

To answer our research questions, extensive experiments were conducted and results were collected and analyzed statistically. Both the algorithms were implemented using MATLAB and run for 30 times on each version of the flex program to overcome the internal threats to validity. Each algorithm was executed for 500 iterations in a single run. So each algorithm was run for a total of 15000 function evaluations for each version of the flex program. Each algorithm used 20 particles i.e. bats and cuckoos as the initial population. Each algorithm was restricted to select only 5 test cases out of 567 test cases. After 30 executions of each algorithm, results were collected and statistically analyzed using F-test to test the research hypothesis. An F-test is used to test if the variances of two populations are equal. Each version of Flex program contains 567 test cases numbered from T1 to T567. Seeded faults are numbered from F1 to F20. Table 1 below shows the characteristics of the subject programs.

Table 1. Characteristics of the Subject Programs

Program Version	Total No. of test cases	Total No. of faults
Flex V1	567	19
Flex V2	567	20
Flex V3	567	17
Flex V4	567	16
Flex V5	567	9

C. Observations and Discussion of Results

Table 2 below displays the average values of the collected metrics.

Table 2. Average Values of the Collected Metrics

Program version	Total No of faults available	Max. No. of faults detected by Bat	Max. No. of faults detected by Cuckoo Search	Avg. No. of faults detected by Bat	Avg. No. of faults detected by Cuckoo Search	Avg. Execution time of Bat	Avg. Execution time of Cuckoo Search
Flex v1	19	16	16	13.90	15.07	61.03	112.51
Flex v2	20	11	12	9.27	11.17	63.50	114.51
Flex v3	17	7	8	4.10	7.03	38.17	108.08
Flex v4	16	11	11	10.93	11.00	37.60	104.36
Flex v5	9	5	5	5.00	5.00	45.58	80.41

Figure 1, 2 and 3 below show the collected metrics in the graphical form.

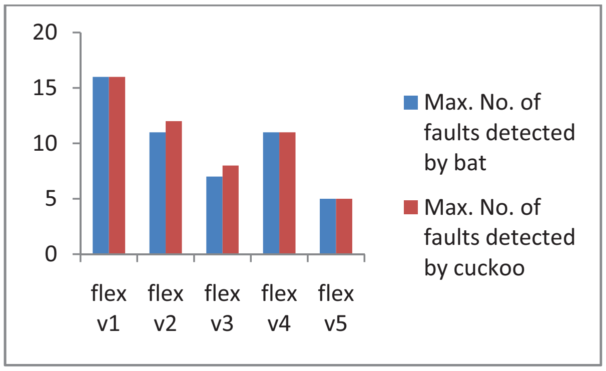

Figure 1. Maximum Number of Faults Detected by Bat and Cuckoo Search Algorithm

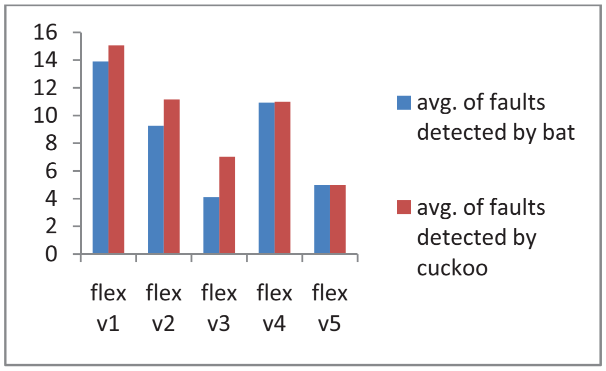

Figure 2. Average No of Fau lts Detected by Bat and Cuckoo Search Algorithm

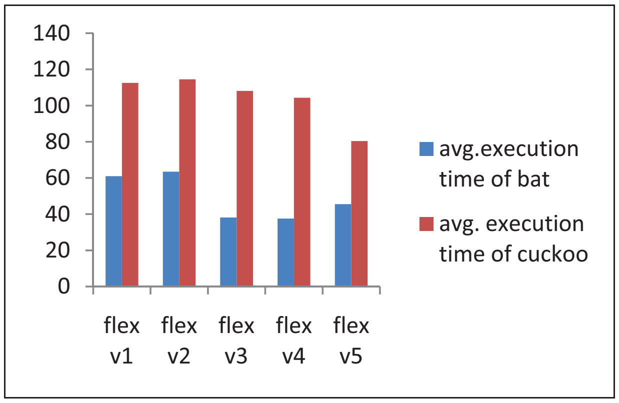

Figure 3. Average Execution Time of Bat and Cuckoo Search Algorithm

Table 3 below shows the descriptive statistics of both the algorithms for the metric number of faults covered obtained from the statistical software SPSS.

Table 3. Descriptive Statistics of Bat and Cuckoo Search Algorithm for No of Faults Covered

Algorithm	N	Mean	Std. Dev.	Std. Error	95% Confidence Interval for Mean		Minimum no of faults covered	Maximum no of faults
					Lower Bound	Upper Bound		
Bat	150	8.64	3.755	0.307	8.03	9.25	3	16

Cuckoo Search	150	9.85	3.534	0.289	9.28	10.42	5	16
Total	300	9.25	3.690	0.213	8.83	9.67	3	16

It is evident from this table that the mean value of the number of faults covered for 30 runs on each version of the flex program for cuckoo search algorithm is more than the mean value for the bat algorithm as well as the standard deviation for the cuckoo search algorithm is lesser for the bat algorithm. Moreover Minimum number of faults covered by cuckoo search algorithm is greater than the bat algorithm. Both these facts indicate that the null hypothesis should be rejected. Table 4 below shows the results of the F-test conducted using SPSS. In our context, the null hypothesis states that regarding the fault coverage, two algorithms perform equally well. Confidence interval α was set at 0.95.

Table 4. F-test Statistics

Source of Variation	Sum of Squares	Df	Mean Square	F	Sig.
Between Groups	110.413	1	110.413	8.306	0.004
Within Groups	3961.333	298	13.293		
Total	4071.747	299			

It is evident from Table 4 that the observed sig. value is 0.004 which is less than the value of α (0.05), so the null hypothesis can be rejected in favor of alternate hypothesis. It means there is statistically significant difference between the performance of bat and cuckoo search algorithms which answers our research question 1. In our case cuckoo search algorithm performs a little better than the bat algorithm in terms of number of faults covered. Table 5 and 6 below show the test cases selected and their corresponding execution times for each version of the flex program for Bat and Cuckoo Search Algorithm respectively. It may be noted that we have considered only those sets of test cases for which the algorithm running time was minimum and the number of faults covered was maximum out of 30 runs. Last row in each table displays the collective execution time required for each set of test cases. However the time required executing the cuckoo search algorithm is far more than the time required to execute the bat algorithm for the same problem and is evident from the values in Tables 5 and 6.

Table 5. Test cases selected by Bat Algorithm and their execution time

Bat Algorithm									
Flex V1		Flex V2		Flex V3		Flex V4		Flex V5	
Test Case	Exec. Time t_i	Test Case	Exec. Time t_i	Test Case	Exec. Time t_i	Test Case	Exec. Time t_i	Test Case	Exec. Time t_i
T163	0.37	T80	0.25	T269	0.36	T78	0.25	T48	0.33
T430	0.37	T10	0.20	T152	0.27	T21	0.19	T60	0.25
T428	0.32	T564	0.31	T46	0.31	T363	0.29	T356	0.33
T216	0.32	T498	0.27	T47	0.36	T204	0.25	T101	0.30
T322	0.32	T249	0.28	T555	0.52	T355	0.35	T354	0.41
Σti	1.70	Σti	1.30	Σti	1.81	Σti	1.34	Σti	1.63

Table 6. T est cases selected by Cuckoo Search Algorithm and their execution time

Cuckoo Search Algorithm									
Flex V1		Flex V2		Flex V3		Flex V4		Flex V5	
Test Case	Exec. Time t_i	Test Case	Exec. Time t_i	Test Case	Exec. Time t_i	Test Case	Exec. Time t_i	Test Case	Exec. Time t_i
T438	0.25	T8	0.22	T9	0.27	T556	0.45	T428	0.32
T413	0.36	T58	0.35	T189	0.26	T525	0.32	T315	0.32
T182	0.27	T10	0.20	T1	0.15	T181	0.25	T106	0.27
T518	0.25	T533	0.32	T304	0.23	T196	0.28	T23	0.23
T338	0.25	T173	0.30	T1	0.15	T277	0.27	T117	0.33
Σti	1.39	Σti	1.40	Σti	1.06	Σti	1.56	Σti	1.46

Table 7 below shows the savings in efforts that can be achieved by Bat and Cuckoo Search algorithms.

Table 7. Savings in Efforts in terms of Execution time

Algorithm	Program Version	Time Required to Execute all 567 test cases T1	Avg. Execution time T2	Time Required to Execute selected 5 test cases T3	Total Time Required (T2 + T3)	Time Saved T1 - (T2 + T3)	%age Time Saved

Bat	Flex V1	170.38	61.03	1.70	62.73	107.64	63.18
	Flex V2	170.38	63.50	1.30	64.80	105.58	61.97
	Flex V3	170.38	38.17	1.81	39.98	130.40	76.54
	Flex V4	170.38	37.60	1.34	38.95	131.43	77.14
	Flex V5	170.38	45.58	1.63	47.21	123.17	72.29
Cuckoo Search	Flex V1	170.38	112.51	1.39	113.90	56.48	33.15
	Flex V2	170.38	114.51	1.40	115.91	54.47	31.97
	Flex V3	170.38	108.08	1.06	109.14	61.24	35.94
	Flex V4	170.38	104.36	1.56	105.92	64.45	37.83
	Flex V5	170.38	80.41	1.46	81.87	88.51	51.95

It is evident from the values in table 7 that around 65% effort can be saved using Bat Algorithm while around 36% time can be saved using cuckoo search algorithm which answers our research question 2. It means that bat algorithm performs better than cuckoo search algorithm from the point of view of execution time. We have taken into consideration both the running time of algorithm as well as the time required to execute the selected test cases.

VII. THREATS TO VALIDITY

Any empirical evaluation such as this suffers from threats to validity that can affect results. This section discusses such threats to validity and how they have been addressed in the study.

To address the construct validity we have used well defined metrics (Number of faults covered and execution time) for investigating whether any algorithm performs significantly different than other. To mitigate such a threat, the number of seeded faults data was taken from the benchmark dataset from SIR repository. Threats to internal validity involve the random nature of the meta-heuristic techniques themselves. In order to ensure reliable comparison and avoid any biases which could affect the obtained results, parameter settings in the experiments were kept uniform and number of generations was set at 500 so that it could not harm or favor the performance of one meta-heuristic over the other. Inherent stochastic behavior of the meta–heuristic search algorithms can be another potential source of bias. To address this source of variation each algorithm was executed for 30 runs. To check the superiority of one algorithm over the other, statistical tests were performed.

Instances used in the case study may cause a threat to the external validity. The selected versions of the subject programs are benchmark instances available from SIR repository, which is well maintained repository widely used in the previous works.

Finally to address the conclusion validity, we support our findings by using proper statistics. Descriptive statistics and F-test available from SPSS was used to reject the null hypothesis.

VIII. CONCLUSION AND FUTURE SCOPE

The main contribution of this paper was to investigate the performance of Bat and Cuckoo Search algorithms to select test cases from a test pool considering both the fault coverage and execution time. We point out that the bat algorithm was investigated for the first time by us in the context of test case selection, to the best of our knowledge. We also point out that the two algorithms can be adapted to other test case selection criteria and are not limited to two objective functions.

Hence we conclude that cuckoo search algorithm performs better than bat algorithm in terms of number of faults covered metric however according to algorithm running time, bat algorithm performs better than cuckoo search algorithm. So there is a tradeoff between the number of faults covered and execution time. In future we wish to perform the same experiments on a large number of programs in order to verify whether the obtained results are equivalent to those presented here. The impact of parameter tuning on the performance of algorithms for test case selection can be investigated in future. We also intend to implement other multiobjective approaches for a more complete comparison among techniques.

IX. REFERENCES

[1] Yang, X.S., 2014. Nature-inspired optimization algorithms. Elsevier.

[2] Nachiyappan, S., Vimaladevi, A. and SelvaLakshmi, C.B., 2010, December. An evolutionary algorithm for regression test suite reduction. In *Communication and Computational Intelligence (INCOCCI), 2010 International Conference on* (pp. 503-508). IEEE.

[3] Rothermel, G., Elbaum, S., Kinneer, A. and Do, H., 2006. Software-artifact infrastructure repository. *UR L http://sir. unl. edu/portal*.

[4] Nakamura, R.Y., Pereira, L.A., Costa, K.A., Rodrigues, D., Papa, J.P. and Yang, X.S., 2012, August. BBA: a binary bat algorithm for feature selection. In *2012 25th SIBGRAPI Conference on Graphics, Patterns and Images* (pp. 291-297). IEEE.

[5] Wang, G. and Guo, L., 2013. A novel hybrid bat algorithm with harmony search for global numerical optimization. *Journal of Applied Ma thematics,2013*.

[6] Khan, K. and Sahai, A., 2012. A comparison of BA, GA, PSO, BP and LM for training feed forward neural networks in e-learning context. *International Journal o f Intelligent Systems and Applications*, *4*(7), p.23.

[7] Fister, I., Rauter, S., Yang, X.S. and Ljubič, K., 2015. Planning the sports training sessions with the bat algorithm. *Neurocomputing*, *149*, pp.993-1002.

[8] Xie, J., Zhou, Y. and Zheng, H., 2013. A hybrid metaheuristic for multiple runways aircraft landing problem based on bat algorithm. *Journal of Applied Mathematics*, *2013*.

[9] Ye, Z.W., Wang, M.W., Liu, W. and Chen, S.B., 2015. Fuzzy entropy based optimal thresholding using bat algorithm. *Applied Soft Computing*, *31*, pp.381-395.

[10] Srivastava, P.R., Bidwai, A., Khan, A., Rathore, K., Sharma, R. and Yang, X.S., 2014. An empirical study of test effort estimation based on bat algorithm. *International Journal of Bio-Inspired Computation*, *6*(1), pp.57-70.

[11] Yildiz, A.R., 2013. Cuckoo search algorithm for the selection of optimal machining parameters in milling operations. *The*

International Journal of Advanced Manufacturing Technology, *64*(1-4), pp.55-61.

[12] Durgun, I. and Yildiz, A.R., 2012. Structural design optimization of vehicle components using cuckoo search algorithm. *Materials Testing*, *54*(3), pp.185-188.

[13] Layeb, A., 2011. A novel quantum inspired cuckoo search for knapsack problems. *International Journal of Bio-Inspired Computation*, *3*(5), pp.297-305.

[14] Ouaarab, A., Belaïd A., and Yang, X.S., 2014. Discrete cuckoo search algorithm for the travelling salesman problem. *Neural Computing an d Applications* 24(7-8), pp. 1659-1669.

[15] Burnwal, S. and Deb, S., 2013. Scheduling optimization of flexible manufacturing system using cuckoo search-based approach. *The International Journal of Advanced Manufacturing Technology*, *64*(5-8), pp.951-959.

[16] Chandrasekaran, K. and Simon, S.P., 2012. Multi-objective scheduling problem: Hybrid approach using fuzzy assisted cuckoo search algorithm. *Swarm and Evolutionary Computation*, *5*, pp.1-16.

[17] Gherboudj, A., Layeb, A. and Chikhi, S., 2012. Solving 0-1 knapsack problems by a discrete binary version of cuckoo search algorithm. *International Journal of Bio-Inspired Computation*, *4*(4), pp.229-236.

[18] Marichelvam, M.K., 2012. An improved hybrid Cuckoo Search (IHCS) metaheuristics algorithm for permutation flow shop scheduling problems. *International Journal of Bio-Inspired Computation*, *4*(4), pp.200-205.

[19] Srivastava, P.R., Varshney, A., Nama, P. and Yang, X.S., 2012. Software test effort estimation: a model based on cuckoo search. *International Journal of Bio-Inspired Computation*, *4*(5), pp.278-285.

[20] Srivastava, P.R., Khandelwal, R., Khandelwal, S., Kumar, S. and Santebennur Ranganatha, S., 2012. Automated test data generation using cuckoo search and tabu search (CSTS) algorithm.

[21] Srivastava, P.R., Sravya, C., Ashima, Kamisetti, S. and Lakshmi, M., 2012. Test sequence optimisation: an intelligent approach via cuckoo search. *International Journal of Bio-Inspired Computation*, *4*(3), pp.139-148.

[22] Nagar, R., Kumar, A., Singh, G.P. and Kumar, S., 2015, February. Test case selection and prioritization using cuckoos search algorithm. In *Futuristic Trends on Computational Analysis an d Knowledge Management (ABLAZE), 2015 International Conference on* (pp. 283-288). IEEE.

[23] Ali, A., Nadeem, A., Iqbal, M.Z.Z. and Usman, M., 2007, December. Regression testing based on UML design models. In *Dependable Computing, 2007. PRDC 2007. 13th Pacific Rim International Symposium on*(pp. 85-88). IEEE.

[24] De Souza, L.S., Prudêncio, R.B. and Barros, F.D.A., Multi-Objective Test Case Selection: A study of the influence of the Catfish effect on PSO based strategies.

[25] Do, H., Elbaum, S. and Rothermel, G., 2005. Supporting controlled experimentation with testing techniques: An infrastructure and its potential impact. *Empirical Software Engineering*, *10*(4), pp.405-435.

[26] Yang, X.S., 2010. A new metaheuristic bat-inspired algorithm. In *Nature inspired cooperative s trategies for optimization (NICSO 2010)* (pp. 65-74). Springer Berlin Heidelberg.

[27] Biswal, S., Barisal, A.K., Behera, A. and Prakash, T., 2013, April. Optimal power dispatch using BAT algorithm. In *Energy Efficient Technologies for S ustainability (ICEETS), 2013 International Conference o n* (pp. 1018-1023). IEEE.

[28] Yang, X.S. and Deb, S., 2010. Engineering optimization by cuckoo search. *International Journal of Mathematical Modelling and Numerical Optimisation*,*1*(4), pp.330-343.

[29] Yang, X.S. and Deb, S., 2009, December. Cuckoo search via Lévy flights. In *Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on* (pp. 210-214). IEEE.