An Effective Approach for Regression Test Case Selection using Pareto based Multi-Objective Harmony Search

Ankur Choudhary
Amity University Uttar Pradesh
P.O. Box 201301
India
ankur.tomer@gmail.com

Arun Prakash Agrawal
Amity University Uttar Pradesh
P.O. Box 201301
India
arun_pr_agrawal@yahoo.co.uk

Arvinder Kaur
USICT,GGSIP University,
P.O. Box 110078
India
arvinderkaurtakkar@yahoo.com

ABSTRACT

Regression testing is a way of catching bugs in new builds and releases to avoid the product risks. Corrective, progressive, retest all and selective regression testing are strategies to perform regression testing. Retesting all existing test cases is one of the most reliable approaches but it is costly in terms of time and effort. This limitation opened a scope to optimize regression testing cost by selecting only a subset of test cases that can detect faults in optimal time and effort. This paper proposes Pareto based Multi-Objective Harmony Search approach for regression test case selection from an existing test suite to achieve some test adequacy criteria. Fault coverage, unique faults covered and algorithm execution time are utilised as performance measures to achieve optimization criteria. The performance evaluation of proposed approach is performed against Bat Search and Cuckoo Search optimization. The results of statistical tests indicate significant improvement over existing approaches.

KEYWORDS

Software testing, Regression testing, Optimization, Harmony Search, Bat Search Optimization, Cuckoo Search Optimization, Test case selection.

1 INTRODUCTION

In present era the software's are the key to success for every industry. The modern competitive environment, technical advancement and ever changing customer requirements are the important factors for software up gradation and maintenance. This maintenance or new software release should go through regression testing because this is the time when new errors are introduced into the previous working release of software. [1]

[1] Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.
SBST'18, May 28–29, 2018, Gothenburg, Sweden
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-5741-8/18/05…$15.00
https://doi.org/10.1145/3194718.3194722

So, to avoid imperfect debugging risks, previous test cases need to be executed again and new test cases should be introduced to ensure the quality of newly added functionality, also called as regression testing [1]. According to National Institute of Standards and Technology (NIST) press release in 2002 the regression testing is a very expensive activity which costs approximately $60 billion every year[10].

Re-execution of all existing test cases is very expensive approach in terms of time and effort, so it becomes important to optimize the regression testing cost by selecting a subset of existing test cases.
The selection of subset of test cases from an existing test suite is an optimization problem [8], which aims to maintain the optimal balance between fault revealing ability, time and effort. Regression test selection (RTS) is an attempt to re-execute a subset of previously executed test cases to ensure that the performed changes do not adversely affect the current software release. Literature revealed that various approaches have been proposed for optimization of regression testing efforts. It is observed that the testing cost or efforts might be reduced by selecting a subset of previously executed test cases with high fault revealing ability [27]. Search based software testing (SBST) is research domain that utilizes metaheuristics for optimization of testing cost as well as optimizing internal factors to maximize fault coverage [4,6,25,27]. Nature has been a source of motivation for us all the time. Nature inspired approaches have been successfully applied in the area of software testing [7,14]. RTS is an NP complete problem in which an optimal balance is required between fault coverage and time. As evident from literature various single as well multi-objective approaches have been formulated to find the optimal solution of RTS. Various nature inspired approaches such as Genetic Algorithm, Cuckoo Search Optimization, and Bat Search Optimization etc. have also been utilized for subset selection in regression testing but these are not the panacea.

In this paper, we propose a regression test case selection strategy using Pareto based multi-objective Harmony Search Algorithm. Harmony search optimization is one of the most explored nature inspired approaches inspired from music. RTS is a combinatorial optimization problem, as it considers a large number of test case combinations to find the optimal subset of test cases. Nature inspired approaches are utilized to solve such type of optimization problems. Pareto solutions are a set of solutions which are nondominated. An optimal Pareto solution should have good solution diversity and convergence. There exist various Pareto based optimizations to achieve the optimal Pareto front. Total no of faults covered, unique faults covered and algorithm execution time are considered as objectives in our problem. Fault coverage and execution time taken by the proposed approach are used as performance measures. The performance of proposed approach is evaluated upon twelve versions of five open source benchmarked experimental objects from software artifact infrastructure repository (SIR) [2]. Results are analyzed and various statistical tests like ANOVA are conducted to validate the results. Results indicate the superiority of Pareto based multi-objective Harmony Search over Cuckoo Search Optimization and Bat Search Algorithm in terms of Average number of faults detected for some experimental objects and comparable performance in others. The rest of paper is organized as: Section 2 discusses related work, Section 3 describes the problem statement, Section 4 describes the proposed work, Section 5 discusses the experimental design and results, Section 6 addresses the threats to validity of our study and finally section 7 concludes the paper.

2 RELATED WORKS

Nature has always been the source of inspiration and learning. Application of nature inspired algorithms for solving complex problems in software testing is known as search based software testing(SBST) [14].

In 1976 Webb Miller and David Spooner first time discussed the term SBST. In 2001, Graves et al. performed a study on various regression test case selection strategies and classified them in five different categories such as: Minimization-based RTS, Dataflow-coverage based RTS, Safe RTS, Ad Hoc-Random RTS and Retest all RTS [5]. In the same year, Harrold et al. also performed empirical Analysis of prediction model for RTS [9]. In 2007, Harman et al. performed an extensive survey on regression test case selection, minimization and prioritization [7]. Yoo et al. proposed a pareto efficient multi-objective test case selection approach [28]. In 2008 Engstrom et al. presented a systematic review on 28 research articles and explored 32 strategies and 15 case studies in the review [3]. In 2009, Maia et al. proposed multi-objective genetic algorithm- NSGA-II for RTS and concluded that NSGA-II based approach performs well in comparison to Random Search [12]. In 2010, Nachiyappan et al. proposed RTS approach using Genetic Algorithm, where coverage and test case execution time are considered as fitness criteria [18]. In 2012, Rothermel et al. proposed a framework for evaluating the regression test case selection [21] techniques. Lucia et al. proposed multi-objective Pareto efficient genetic algorithm to perform optimization of code coverage and computational cost [11]. Mirarab et al. proposed Size-constrained RTS using multi-criteria optimization by utilizing Integer Linear Programming approach [15]. In 2014, Souza et al. proposed an approach using binary particle swarm optimization for RTS [23]. Musa et al. proposed an approach for RTS and prioritization of test cases using Dependence Graph and Genetic Algorithm [17]. Sarker et al. published and discussed the important aspects to select the appropriate optimization technique [22]. Souza et al. proposed a multi-objective test case selection using PSO and Catfish inspired approach [23]. Mondal et al. presented and discussed heuristic based multi-objective test case selection strategy and analyzed it's performance on real world case study such as Apache Ant, Derby, JBoss, nano-XML and Math [16]. Wang et al. proposed an approach for test case minimization for product line followed by elimination of redundant test cases [26]. A major drawback of traditional RTS approaches is that they do not consider available time to retest as an input parameter while the cost of any RTS technique is justifiable if it can reveal maximum number of faults within a given time frame by executing only a subset of test cases. The biggest hurdle in solving time aware RTS problems is non availability of execution times of individual test cases.

To address RTS problem, various authors have considered various factors to find optimal balance between - such as fault coverage, execution time, code coverage etc. Both heuristic and metaheuristics based approaches for RTS exist in the literature. As already stated, nature have inspired the researchers to solve the complex optimization problems. Various nature inspired algorithms have been utilised to perform RTS. Few researchers have applied single objective metaheuristics such as Bat Search, cuckoo search, ant colony optimization and hybrid particle swarm optimization by considering the fault coverage as fitness indicator. Multi-objective metaheuristics such as multi-objective genetic algorithm, NSGA-II, particle swarm optimization, hybrid PSO etc. are also successfully applied to RTS by considering two factors such as fault coverage and code coverage [12,13,15,16,19,24]. But scope of optimization still persists. To attempt RTS problem we have considered two important factors total fault coverage and unique faults covered. There should be a good balance between total fault coverage and unique faults covered. To achieve equilibrium between these we have utilized Pareto-based Multi-Objective Harmony Search.

3 PROBLEM STATEMENT

In RTS, the literature reveals that a variety of approaches differing in selection criteria have been proposed so far, but most of them are based on surrogate measure code coverage and are single objective. However multi objective test case selection approach is promising but it has not been explored to a large extent and in most of the cases code coverage and test case execution time were considered.

In multi-objective optimizations, optimization criteria are generally contradictory to each other and the simplest way to solve these problems is to combine these contrasting optimization criteria into a single objective function by assigning them some weights and finding equilibrium among them. Search based algorithms are used to solve RTS problems, by finding optimal or near optimal solutions with respect to objective function. It is evident from literature that the performance of search based algorithms decreases as the number of optimization criteria increases and it happens due to loss of diversity [11]. These algorithms do not guarantee the diversity of the generated solutions and the algorithm may get stuck into local optima. An increase in diversity increases the ability to detect faults. Moreover, relative performance of different techniques varies across different programs, leaving researchers to develop an approach that shows consistency in results [15].

In RTS, we wish to cover maximum number of faults and maintain the balance between redundant and unique fault revealing test cases out of the subset test suite, thereby reducing the effort required for retesting the modified version of the software under test. In our case, the parameter unique faults covered, allows us to take care of the diversity between solutions (candidate test suites) generated by harmony search. A subset x is considered pareto-optimal if and only if there is no any subset y which can increase the value of one objective function without decreasing the others. Utilizing past fault coverage history of test cases, we can define the problem as follows:

Given: Let P be a program and P' be a modified version of the program P. $T = \{T1, T2, \dots\dots, Tn\}$ is the test suite to test P. Let $n' \leq n$ is a number given and Fn is a function from T to a positive scalar quantity.

Problem: Find S subset of T such that |S| = n' and for all S' subset of T, |S'| = n' implies F(S') ≤ F(S).
F is a score function that assigns a scalar quantity to S', the sum of number of faults detected by S' and the unique faults covered by elements of S'.
Let's have a fault matrix FM of size m*n in the following format where columns represent the test cases and rows represent the faults and each element x_{ij}, where $x_{ij} = \{0, 1\}$, denotes i^{th} fault identified by j^{th} test case.

	t_1	t_2	t_3	t_4	t_5	……	t_{n-1}	t_n
f_1	1	0	0	1	1		0	0
f_2	0	1	0	0	1		1	0
f_3								
.								
f_m	0	1	1	0	0		1	0

If $x_{ij} = 1$ then it indicates that fault i is covered by test case j otherwise not.
We can derive the following from the matrix FM_{m*n}:
F: Set of all faults {f1, f2, f3…. fm}
TS: Set of all test cases i.e. $\{t_1, t_2, t_3, t_4 \dots\dots, t_n\}$
S: Set of selected test cases in any candidate solution.
UFC: represents the uncovered fault count.
Tid_j: j^{th} test case in S covering a unique fault i.
UF: unique faults identified by selected test suite S

Regression Test Case Selection (RTS) can be described as: Given a fault matrix $FM_{m*n} = (x_{ij})$ where $x_{ij} = \{0, 1\}$, the objective of Regression Test Case Selection (RTS) is to select a few columns of matrix FM with minimum cost such that each fault i is covered by at least one selected test case j. Mathematically RTS can be formulated as below:

Minimize $UFC = \sum_{i=1}^{m} f_i - \sum_{j=1}^{n\prime} Tc_j$ (1)
Maximize $UF = \sum_{i=1}^{n\prime} Tid_i$ (2)

In this paper, we have considered three variants of regression test case selection problem as RTS_5, RTS_10 and RTS_20 where 5, 10 and 20 represent the number of selected test cases i.e. test suite size.

4 PROPOSED WORK

Harmony Search algorithm was coined by Xin-She Yang in 2009 as metaheuristic, based on music improvisation to obtain the perfect harmony state. [4]. Exploration and exploitation are achieved in harmony search optimization using following parameters:

Memory Size of Harmony (MSH)
Memory Considering Rate of Harmony (MCRH)
Harmony Bandwidth (HBW)
Harmony Pitch Adjustment Rate (HPAR).

Optimal tuning of these parameters is the key to obtain global optimal solution.
In proposed work, we have tuned MCRH and MSH parameters utilizing an orthogonal array approach and Taguchi approach to find tune the parameters and obtain their optimal values to conduct experiments.
The proposed approach for RTS using multi-objective HS is as follows:
Define both the fitness functions: Minimize: $\mathrm{Obj}_{\mathrm{Fun1}(UFC)}$, Where UFC stands for uncovered faults count.
Maximize: $\mathrm{Obj}_{\mathrm{Fun2}(UF)}$, where UF stands for unique faults covered, the pareto optimization and harmony search algorithm is utilized to find a balance between both the objectives. $\sum_{i=1}^{M}\sum_{j=1}^{N} TC_i^j \in TC_1^1, TC_1^2, \dots, TC_2^1, TC_2^2, \dots . TC_M^N$, TC is a subset of test cases in which every solution TC_i contains a subset of test suite and i varies from 1-M (MSH), while TC_i^j is an individual test case and j varies from 1-N (number of test cases in subset). According to our problem, different test suites have different test cases.

I. Initialize the pareto list and value of w (balancing factor) where
$\mathrm{Obj}_{\mathrm{Fun(TS)}} = \mathrm{Obj}_{\mathrm{Fun1}(UFC)} * \mathrm{w} + \mathrm{Obj}_{\mathrm{Fun2}(UF)} * (1 - \mathrm{w})$
II. The HS optimization algorithm parameters: MSH and MCRH are initialized as suggested by Taguchi Method and

the maximum number of iteration (Maxitr) is selected as 1000.

III. The Harmony Memory matrix (HMM) of size [HMS x N] is randomly initialized. The random values are selected from the original test suite T. Here ith row represents a candidate solution and jth column represents the subset test suite as $TC_{i=1-HMS}^{j}$

IV. Following steps are utilized to generate the improvised harmony according to following steps:
For each unknown test case from j = 1 to N

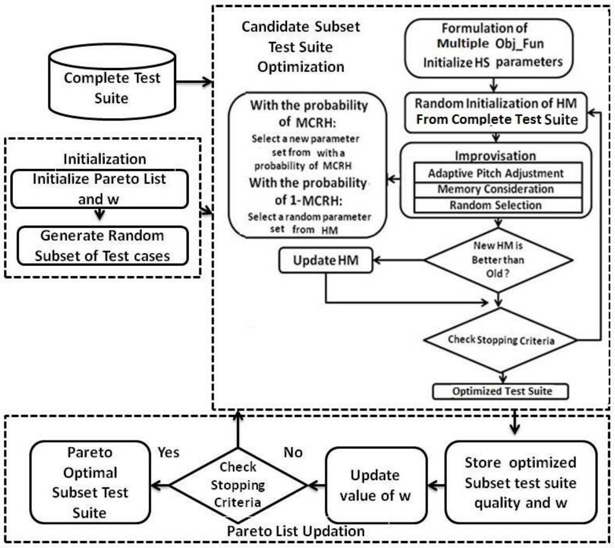
Complete Test Suite
Candidate Subset Test Suite Optimization
Formulation of Multiple Obj_Fun Initialize HS parameters
Random Initialization of HM From Complete Test Suite
With the probability of MCRH: Select a new parameter set from with a probability of MCRH
With the probability of 1-MCRH: Select a random parameter set from HM
Improvisation
Adaptive Pitch Adjustment
Memory Consideration
Random Selection
Initialization
Initialize Pareto List and w
Generate Random Subset of Test cases
Update HM
New HM is Better than Old?
Check Stopping Criteria
Optimized Test Suite
Pareto Optimal Subset Test Suite
Yes
No
Check Stopping Criteria
Update value of w
Store optimized Subset test suite quality and w
Pareto List Updation

Figure 1: Proposed Approach Workflow

$$HBW_{gen} = HBW_{max} exp(c.gen) \tag{2}$$

$$c = \frac{Ln(\frac{HBW_{min}}{HBW_{max}})}{SN} \tag{3}$$

$$tc^{j'} \leftarrow tc^{j'} \pm rand\ (0,1) * HBW \tag{4}$$

Where $HPAR_{min}$ and $HPAR_{max}$ are low and high value of HPAR respectively. Gen represents the number of generations. $HPAR_{gen}$ denotes the HPAR per generation gen. SN is solution index in solution vector per generation. HBW_{min} and HBW_{max} are low and high values of HBW respectively. HBW_{gen} denotes the HBW for generation gen.

V. The worst solution in HM will be replaced by newly generated solution $TC^{j'}$, if the new solution is better than worst solution.

VI. Repeat steps III and V for harmony improvisation and HM update until the stopping criterion is met.

VII. Repeat steps III and VI for Pareto optimal solution until the stopping criterion is met.

5 EXPERIMENTS AND RESULT DISCUSSION

This section presents the experiments conducted and analysis of results for performance evaluation of proposed, Bat, and Cuckoo

a. With a probability of 1-MCRH, generate the new solution $TC^{j'}$ from the original test suite T.

b. With a probability of 1-MCRH, randomly pick $TC^{j'}$ from the jth column of the HMM.

c. The selected $TC^{j'}$ will undergo pitch adjustment using Equation (1) to (4).

$$HPAR_{gen} = HPAR_{min} + \frac{(HPAR_{max} - HPAR_{min})}{SN} * gen \tag{1}$$

Search approaches applied to RTS problem. Section 5.1 and 5.2 discuss briefly the research objectives and the subject programs used in the study. Section 5.3 presents the control parameter settings of the search algorithms used after fine tuning of parameter values using Taguchi method. Section 5.4 presents the research hypotheses formed and results obtained after statistical tests conducted. Results have been analyzed and discussed in section 5.5.

5.1 Research Objectives

We attempt to answer the following research questions through this study:

RQ1. How well do the search algorithms used in study perform in comparison to each other?

RQ2. What is the effect of different test suite sizes on fault revealing capabilities of three approaches used in the study?

5.2 Subject Programs

Five versions of lexical analyzer flex, one version of grep, gzip and xml-security each and four versions of nano xml written in c and java have been retrieved from benchmarked software artifact infrastructure repository (SIR) [2]. Fault matrices of the subject programs under study were retrieved using shell scripts from SIR. Characteristics of the subject programs are presented in Table 1 below:

Table 1: Characteristics of Subject Programs

Objects	Total number of seeded faults	Total number of test cases	Test suite type
flex v1	19	567	TSL
flex v2	20	567	TSL
flex v3	17	567	TSL
flex v4	16	567	TSL
flex v5	9	567	TSL
Grep	18	199	TSL
Gzip	16	214	TSL
nano-xml v1	5	15	JUnit
nano-xml v2	7	214	JUnit
nano-xml v3	7	216	JUnit
nano-xml v5	9	216	JUnit
xml-security	5	15	JUnit

5.3 Parameter Tuning of Search Algorithms

It is always good to conduct experiments on those values of algorithm parameters at which we get the best results. Taguchi method [20] was used to fine tune the parameters of the algorithms with different parameter value settings by running each experiment 30 times to remove any bias and 500 iterations in each run. Following parameter value settings were found to be optimal to conduct the further experiments as shown in Table 2 below:

Table 2: Optimal Parameter Values of Algorithms

Optimal Parameter Values for HS Algorithm				**Optimal Parameter Values for Bat Algorithm**				**Optimal Parameter Values for Cuckoo Search**	
HMS	MCRH	HBW	HPAR	No. of Bats	Frequency	Loudness	Pulse Rate	No. of Nests	Probability that a host bird will discover alien egg
50	0.9	0.30	0.30	40	75	0.75	0.25	20	0.5

5.4 Research Hypotheses and Results Obtained

We have developed the following two research hypotheses to answer our research questions 1 and 2 respectively:

Ho: BAT = CS = HS
Ha: BAT ≠ CS ≠ HS

Ho: Effectiveness of RTS_5 = Effectiveness of RTS_10 = Effectiveness of RTS_20
Ha: Effectiveness of RTS_5 ≠ Effectiveness of RTS_10 ≠ Effectiveness of RTS_20

Controlled experiments were conducted by executing the three search algorithms 30 times each with the same optimal parameter value settings for each of the subject program version to gather the values of performance metrics- No. of faults detected and execution time. 500 generations/iterations in each run are used as the stopping criteria for all RTS_5, RTS_10 and RTS_20 problems in case of Bat and Cuckoo Search Algorithms and 1000 iterations for Pareto-based Harmony Search Algorithm.

Table 3 and 4 in the appendix below present the average no. of faults detected and average execution times of three search algorithms for different subject programs and different test suite sizes. Table 5 in appendix presents the maximum number of faults detected data by different algorithms with respect to maximum detectable faults information available from past fault history of subject programs available from SIR.

In order to test our first research hypothesis to answer our RQ1, Table 6 below presents the results of two way ANOVA conducted at 95% confidence interval (α = 0.05) on Number of faults detected by Bat, Cuckoo and Harmony Search algorithms for test suite size of 5 test cases.

Table 6: Results of Two-way ANOVA conducted on Number of Faults detected by Bat, Cuckoo and HS for RTS_5

Source of Variation	**SS**	**df**	**MS**	**F**	**P-value**	**F crit**
Sample (Algorithm)	490.1241	2	245.062	333.03	1.3E-112	3.004345
Columns (Subject program)	11411.39	11	1037.399	1409.786	0	1.797808
Interaction	635.0315	22	28.86507	39.22653	4.5E-120	1.552305
Within	768.2333	1044	0.735856			
Total	13304.78	1079				

To see the effectiveness of test suite size, we conducted two-way ANOVA at 95% confidence interval (α = 0.05) on Number of faults detected by Pareto-based Harmony Search algorithm for different test suite sizes, results of which are presented in Table 7 below:

Table 7: Two-way ANOVA conducted on Number of faults detected by HS for different test suite sizes

Source of Variation	***SS***	***df***	***MS***	***F***	***P-value***	***F crit***
Sample (Test Suite Size)	492.607	2	246	141	7.76E-54	3.006071
Columns (Subject program)	10316.7	9	1146	657	0	1.890625
Interaction	176.104	18	9.78	5.61	1.01E-12	1.615722
Within	1517.47	870	1.74			
Total	12502.8	899				

5.5 Results Discussion and Answers to Research Questions

If we look at the observed F values in the table 6 and 7 above, which are more than the F-critical values as well as the p-values are less than the significance level 0.05, it suggests the rejection of null hypotheses in favor of alternate hypotheses. These results help us in answering our research question 1 and we can say that the performance of at least one approach is statistically significantly different from other two approaches. This observation is also supported by the results of Table 3 in appendix which reflect that the average number of faults detected by Pareto-based Harmony Search are either highest or comparable to other two approaches irrespective of the subject program or the test suite size.

Similarly from the results of table 7, we can say that the effectiveness in terms of fault revealing capability of one test suite size is significantly different from other two. It can also be verified from results of Table 5 in appendix which reflects that the test suite size of 5 test cases is sufficient to detect the maximum number of detectable faults in most of the cases irrespective of the approach used. We can observe from the results of table 5 that there is no significant improvement in the number of faults detected, if we increase the size of the test suite from 5 except of Cuckoo Search.

Results depicted in table 8 below are in favor of proposed Pareto-based Harmony Search algorithm from the perspective of percentage of cases where proposed approach is able to detect maximum coverable faults with respect to different test suite sizes (%age calculated on the basis of max. Faults coverable and max faults covered for each subject program).

Table 8: Performance Comparison of Bat, Cuckoo and Pareto Based Harmony Search Approaches

Algorithm	5 TEST CASE	10 TEST CASE	20 TEST CASE
BAT	66.67%	58.33%	50.00%
CUCKOO	58.33%	91.67%	80%
HS	100%	100%	100%

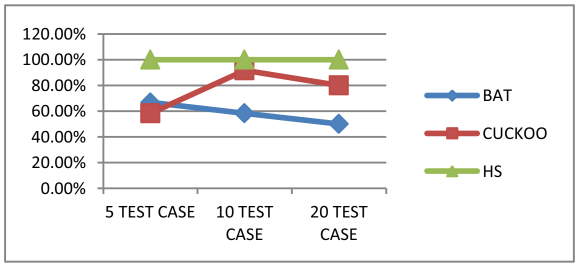

Figure 2: Performance Comparison of approaches with respect to test suite size

We can also observe from the data in table 8 above that there is a decline in the performance of Bat Search algorithm and improvement in the performance of Cuckoo Search algorithm as we increase the size of the selected test suite, while the performance of proposed Pareto-based Harmony Search is consistent. This trend can also be presented graphically as in the figure 2 above. Average execution times, for different search algorithms for different test suite sizes, depicted in table 4 in appendix are also in favour of the proposed Pareto-based harmony search algorithm at a test suite size of five with minimum values for almost all the subject programs used in the study.

6 THREATS TO VALIDITY

Threats to the validity of performance evaluation of adopted RTS approaches are discussed in this section. Relationship between theory and observation is governed by construct validity. In our study, they are related to the performance measures used to evaluate different test case selection algorithms. We have used well known and widely used performance measures by previous researchers. In particular we used: (i) fault coverage (ii) unique faults covered and (iii) execution time of algorithm to build our objective functions.

Stochastic nature of the search algorithms used in the study is an internal threat to validity and may influence the results. To address this problem, 30 runs of each algorithm were executed with 500 generations/iteration in each run and average values of the measures were used for the evaluation purpose to remove any bias. Tuning of algorithm parameters is another source of internal validity, which was addressed by first fine tuning of algorithm parameters through Tagachi Method and then same parameter values were used to conduct further controlled experiments.

Threats to external validity are concerned with the generalization of results obtained and the set of subject programs used for experimentation. We have considered 12 versions of five different subject programs which have also been widely used by a number of previous researchers on regression testing. To address the conclusion validity, we conducted appropriate statistical tests like two-way ANOVA to support our findings.

7 CONCLUSION AND FUTURE WORK

In this paper, we proposed a Pareto based harmony search algorithm for regression test case selection and was performance evaluated empirically with Bat and Cuckoo Search Algorithms through benchmarked subject programs retrieved from SIR Dataset. We then investigated the effect of various test suite sizes in a specific form of regression test case selection problem by taking the number of test cases to be selected as input to address the problem of limited availability of time during regression testing. It is evident from the results that our proposed approach outperformed the other two approaches in most of the cases, statistically tied in others and was rarely outperformed. Consistency of the proposed approach is another important observation in comparison to Bat and Cuckoo Search.

We observed that the test suite size of five test cases was quite enough to meet the test adequacy criteria for most of the programs under study and there is no significant improvement in fault coverage if we increase the test suite size from 5.

In future, we wish to extend the idea presented in this paper by studying in more depth. We would like to replicate the study with other subject programs with real faults, other test suite sizes, evaluation metrics and other optimization algorithms like NSGA II or formulations of the problem which could produce better results.

REFERENCES

[1] David Binkley and Ieee Computer Society. 1997. Test Cost Reduction. 23, 8 (1997), 498–516.

[2] Hyunsook Do, Sebastian Elbaum, and Gregg Rothermel. 2005. Supporting controlled experimentation with testing techniques: An infrastructure and its potential impact. *Empir. Softw. Eng.* 10, 4 (2005), 405–435. DOI:https://doi.org/10.1007/s10664-005-3861-2

[3] Emelie Engström, Mats Skoglund, and Per Runeson. 2008. Empirical evaluations of regression test selection techniques: a

systematic review. *Int. Symp. Empir. Softw. Eng. Meas.* (2008), 22–31. DOI:https://doi.org/10.1145/1414004.1414011

[4] Juan Pablo Galeotti, Gordon Fraser, and Andrea Arcuri. 2014. Extending a search-based test generator with adaptive dynamic symbolic execution. *Proc. 2014 Int. Symp. Softw. Test. Anal. - ISSTA 2014* (2014), 421–424. DOI:https://doi.org/10.1145/2610384.2628049

[5] Todd L Graves, Jung-Min Kim, and Adam Porter. 2001. An Empirical Study of Regression Test Selection Techniques. *ACM Trans. Softw. Eng. Methodol.* 10, 2 (2001), 184–208. DOI:https://doi.org/10.1145/367008.367020

[6] Florian Gross, Gordon Fraser, and Andreas Zeller. 2012. Search-based system testing: high coverage, no false alarms. *Proc. 2012 Int. Symp. Softw. Test. Anal. - ISSTA 2012* (2012), 67. DOI:https://doi.org/10.1145/2338965.2336762

[7] Mark Harman, Yue Jia, and Yuanyuan Zhang. 2015. Achievements , Open Problems and Challenges for Search Based Software Testing. *Proc. 8th IEEE Int. Conf. Softw. Testing, Verif. Valid.* Icst (2015), 1–12. DOI:https://doi.org/10.1109/ICST.2015.7102580

[8] M.J. Harrold and M.L. Souffa. 1988. An incremental approach to unit testing during maintenance. *Proceedings. Conf. Softw. Maintenance, 1988.* (1988), 362–367. DOI:https://doi.org/10.1109/ICSM.1988.10188

[9] Mary Jean Harrold, Ieee Computer Society, David Rosenblum, Senior Member, Gregg Rothermel, Ieee Computer Society, Elaine Weyuker, and Senior Member. 2001. Empirical Studies of a Prediction Model for Regression Test Selection. 27, 3 (2001), 248–263.

[10] Rafaqut Kazmi, Dayang N. A. Jawawi, Radziah Mohamad, and Imran Ghani. 2017. Effective Regression Test Case Selection. *ACM Comput. Surv.* 50, 2 (2017), 1–32. DOI:https://doi.org/10.1145/3057269

[11] Andrea De Lucia, Massimiliano Di Penta, Rocco Oliveto, and Annibale Panichella. 2012. On the role of diversity measures for multi-objective test case selection. *2012 7th Int. Work. Autom. Softw. Test, AST 2012 - Proc.* (2012), 145–151. DOI:https://doi.org/10.1109/IWAST.2012.6228983

[12] Camila Loiola Brito Maia, Rafael Augusto Ferreira Do Carmo, Fabrício Gomes De Freitas, Gustavo Augusto Lima De Campos, and Jerffeson Teixeira De Souza. 2009. A Multi-Objective Approach For The Regression Test Case Selection Problem. *XLI Brazilian Symp. Oper. Res. XLI SBPO 2009.* (2009), 1824–1835. Retrieved from http://sobrapo.org.br/simposios/XLI-2009/XLI_SBPO_2009_artigos/artigos/56096.pdf

[13] Alessandro Marchetto and Mahfuzul Islam. 2013. A Multi-Objective Technique for Test Suite Reduction. *ICSEA 2013, Eighth ...* c (2013), 18–24. Retrieved from http://www.thinkmind.org/index.php?view=article&articleid=icsea_2013_1_30_10119

[14] Phil McMinn. 2011. Search-Based Software Testing: Past, Present and Future. In *2011 IEEE Fourth International Conference on Software Testing, Verification and Validation Workshops*, 153–163. DOI:https://doi.org/10.1109/ICSTW.2011.100

[15] Siavash Mirarab, Soroush Akhlaghi, and Ladan Tahvildari. 2012. Size-constrained regression test case selection using multicriteria optimization. *IEEE Trans. Softw. Eng.* 38, 4 (2012), 936–956. DOI:https://doi.org/10.1109/TSE.2011.56

[16] Debajyoti Mondal, Hadi Hemmati, and Stephane Durocher. 2015. Exploring test suite diversification and code coverage in multi-objective test case selection. *2015 IEEE 8th Int. Conf. Softw. Testing, Verif. Validation, ICST 2015 - Proc.* (2015). DOI:https://doi.org/10.1109/ICST.2015.7102588

[17] Samaila Musa, Abu Bakar MD Sultan, Abd Azim Bin Abdul Ghani, and Salmi Baharom. 2014. Regression Test Case Selection &PrioritizationUsing Dependence Graph and Genetic Algorithm. *IOSR J. Comput. Eng.* IV, 3 (2014), 2278–661.

[18] S Nachiyappan, A Vimaladevi, C B Selvalakshmi, and SelvaLakshmi CB. 2010. An evolutionary algorithm for regression test suite reduction. *Proc. 2010 Int. Conf. Commun. Comput. Intell. (INCOCCI).* (2010), 503–508.

[19] Annibale Panichella, Rocco Oliveto, Massimiliano Di Penta, and Andrea De Lucia. 2015. Improving multi-objective test case selection by injecting diversity in genetic algorithms. *IEEE Trans. Softw. Eng.* 41, 4 (2015), 358–383. DOI:https://doi.org/10.1109/TSE.2014.2364175

[20] J.S. Ramberg, S.M. Sanchez, P.J. Sanchez, and L.J. Hollick. Designing simulation experiments: Taguchi methods and response surface metamodels. In *1991 Winter Simulation Conference Proceedings.*, 167–176. DOI:https://doi.org/10.1109/WSC.1991.185612

[21] G. Rothermel and M.J. Harrold. 2012. A framework for evaluating regression test selection techniques. *Proc. 16th Int. Conf. Softw. Eng.* April (2012), 201–210. DOI:https://doi.org/10.1109/ICSE.1994.296779

[22] Ruhul A Sarker, Saber M Elsayed, and Tapabrata Ray. 2014. Selection for Optimization Problems. *Ieee Trans. Evol. Comput.* 18, 5 (2014), 689–707. DOI:https://doi.org/10.1109/TEVC.2013.2281528

[23] Luciano S De Souza and Ricardo B C Prudˆ. 2014. Multi-Objective Test Case Selection : A study of the influence of the Catfish effect on PSO based strategies. (2014), 3–58.

[24] G Sun and A Zhang. 2013. A hybrid genetic algorithm and gravitational search algorithm for image segmentation using multilevel thresholding. *Iber. Conf. Pattern Recognit. Image* (2013). Retrieved April 29, 2017 from http://link.springer.com/chapter/10.1007/978-3-642-38628-2_84

[25] Mattia Vivanti, Andre Mis, Alessandra Gorla, and Gordon Fraser. 2013. Search-based data-flow test generation. *2013 IEEE 24th Int. Symp. Softw. Reliab. Eng. ISSRE 2013* (2013), 370–379. DOI:https://doi.org/10.1109/ISSRE.2013.6698890

[26] Shuai Wang, Shaukat Ali, and Arnaud Gotlieb. 2015. Cost-effective test suite minimization in product lines using search techniques. *J. Syst. Softw.* 103, C (2015), 370–391. DOI:https://doi.org/10.1016/jjss.2014.08.024

[27] S Yoo and M Harman. 2007. Regression Testing Minimisation, Selection and Prioritisation : A Survey. *Test. Verif. Reliab* 0, (2007), 1–7. DOI:https://doi.org/10.1002/000

[28] Shin Yoo and Mark Harman. 2007. Pareto efficient multi-objective test case selection. *Proc. 2007 Int. Symp. Softw. Test. Anal. - ISSTA '07* (2007), 140. DOI:https://doi.org/10.1145/1273463.1273483

Appendix

Table 3: Average No. of Faults detected by different search algorithms for different test suite sizes

Avg. No. of Faults detected	*size/object*	*flex v1*	*flex v2*	*flex v3*	*flex v4*	*flex v5*	*grep*	*gzip*	*nano v1*	*nano v2*	*nano v3*	*nano v5*	*xml security*
HS	RTS_5	15	12	9	11	5	3	6	5	6	6	8	5
	RTS_1	15	12	7	11	5	3	6	5	6	6	7	5

	0												
	RTS_20	14	11	6	10	5	3	5	n/a	4	5	5	n/a
BAT	RTS_5	14	9	3	10	5	2	3	5	6	6	6	3
	RTS_10	14	10	3	10	5	2	3	4	3	5	5	3
	RTS_20	13	9	4	10	5	1	2	n/a	4	4	3	n/a
CUCKOO	RTS_5	13	11	5	9	5	2	5	3	5	5	6	4
	RTS_10	14	11	5	9	5	3	5	4	5	4	6	4
	RTS_20	13	11	4	8	5	2	3	n/a	3	4	5	n/a

Table 4: Average Execution times of different search algorithms for different test suite sizes

			flex v1	*flex v2*	*flex v3*	*flex v4*	*flex v5*	*Grep*	*Gzip*	*nano xml v1*	*nano xml v2*	*nano xml v3*	*nano xml v5*	*xml security*
AVERAGE EXECUTION TIME	RTS_5	BAT	24.56	24.67	23.16	22.6	19.11	22.19	21.23	15.91	16.74	16.38	17.3	15.41
		CUCKOO	22.76	23.1	21.51	21.13	17.44	20.21	19.19	13.56	14.51	14.49	15.46	13.6
		HS	7	7	7	7	7	7	7	7	7	7	7	7
	RTS_10	BAT	32.73	33.77	30.88	29.86	23.19	28.83	27.05	17.22	19.1	19.19	20.78	17.33
		CUCKOO	30.4	30.94	28.16	27.32	20.89	27.09	25.04	15.56	17.25	17.26	18.94	15.48
		HS	7	7	7	7	7	7	7	7	7	7	7	7
	RTS_20	BAT	45.66	49.61	44.45	42.85	32.8	43.94	40.79	n/a	24.76	24.92	28.22	n/a
		CUCKOO	45.15	46.39	41.49	39.83	28.06	39.16	36.04	n/a	22.09	22.01	25.34	n/a
		HS	7	8	8	8	7	8	8	n/a	7	7	7	n/a

Table 5: Maximum Faults Coverable and Maximum Faults Covered by different algorithms for different test suite sizes

Subject Program		*flex v1*	*flex v2*	*flex v3*	*flex v4*	*flex v5*	*Grep*	*gzip*	*nano v1*	*nano v2*	*nano v3*	*nano v5*	*xml security*
Total faults seeded		19	20	17	16	9	18	16	5	7	7	9	5
Max faults coverable		16	14	9	11	5	3	7	5	6	6	8	5
RTS_5	BAT	16	11	5	11	5	3	4	5	6	6	6	5
	CUCKOO	15	12	6	11	5	3	5	5	6	6	6	5
	HS	16	14	9	11	5	3	6	5	6	6	8	5
RTS_10	BAT	16	12	6	11	5	2	5	5	6	6	6	5
	CUCKOO	16	13	8	11	5	3	6	5	6	6	8	5
	HS	16	13	9	11	5	3	6	5	6	6	8	5
RTS_20	BAT	16	11	7	11	5	3	4	n/a	6	6	6	n/a
	CUCKOO	16	13	7	11	5	3	6	n/a	6	6	8	n/a
	HS	16	14	8	11	5	3	6	n/a	6	6	8	n/a