Value based PSO Test Case Prioritization Algorithm

Erum Ashraf
Department of Computer Sciences
Bahria University, Islamabad
Pakistan

Tamim Ahmed Khan
Department of Software Engineering
Bahria University, Islamabad
Pakistan

Khurrum Mahmood
Department of Computer Sciences,
Bahria University, Islamabad,
Pakistan

Shaftab Ahmed
Department of Software Engineering
Bahria University, Islamabad
Pakistan

***Abstract*—Regression testing is performed to see if any changes introduced in software will not affect the rest of functional software parts. It is inefficient to re-execute all test cases every time the changes are made. In this regard test cases are prioritized by following some criteria to perform efficient testing while meeting limited testing resources. In our research we have proposed value based particle swarm intelligence algorithm for test case prioritization. The aim of our research is to detect maximum faults earlier in testing life cycle. We have introduced the combination of six prioritization factors for prioritization. These factors are customer priority, Requirement volatility, implementation complexity, requirement traceability, execution time and fault impact of requirement. This combination of factors has not been used before for prioritization. A controlled experiment has been performed on three medium size projects and compared results with random prioritization technique. Results are analyzed with the help of average percentage of fault detection (APFD) metric. The obtained results showed our proposed algorithm as more efficient and robust for earlier rate of fault detection. Results are also revalidated by proposing our new validation equation and showed consistent improvement in our proposed algorithm.**

Keywords*—Test case prioritization (TCP); Particle swarm optimization (PSO); Average percentage of fault detection (APFD); Value based software engineering (VBSE)

I. INTRODUCTION

Regression testing is the process of testing software after any functional or non-functional changes. Regression testing ensures that the changes have not affected rest of its modules. The cost and limitation of resources greatly affect regression testing. There are various techniques to cut down the cost of regression testing. One popular approach is to select some of the test cases randomly from the entire testing range but it is not a wise option when high quality software is required [20]. Another feasible option is test case prioritization technique that involves the reordering test cases in a way to achieve certain goal. These goals can be achieving maximum code coverage or to expose maximum faults in earlier time or reduction of cost. Test case prioritization implies eliminates the need to run the entire test case set and only some selective test cases achieve the goals required.

Test case prioritization is a mechanism by which we can rearrange test cases with an intent that allows us to do the prioritization. However, prioritization is NP complete problem in software testing domain and such kind of problems can be efficiently solved by population based stochastic optimization technique (PSO). In 1995 Kennedy and Eberhart propose a population based stochastic optimization algorithm known as particle swarm optimization. We can solve a range of functional optimization problems using PSO and in many cases, it is favorable to use PSO for its fast convergence ability. This ability also distinguishes it from many other global optimization algorithms [5].

We propose a test case prioritization technique using PSO such that we implement PSO as value based test case prioritization technique. We propose to achieve our goal by an earlier fault detection using value based test case prioritization. We use six factors for value based prioritization that include; 1) customer priority 2) implementation complexity 3) requirement volatility 4) requirement traceability 5) fault impact of requirement and 6) execution time. We use first three out of the six factors for new test cases while the rest three are concerned with reusable test cases. Our goal is to set a priority of the test cases to the new best positions so that to expose maximum faults earlier in testing life cycle. We also use average percentage of fault detection (APFD) metric has been used for evaluating the propose value based test prioritization algorithm [5].

Our paper is organization as follows. We explain previous work in Section 2 and devote Section 3 to explain PSO Algorithm in a brief manner. We describe the proposed approach for test case prioritization using PSO in Section 4. We explain algorithm and evaluations in Section 5 and we discuss our experimental results in Section 6. We finally present conclusion and future work in Section 7.

II. RELATED WORK

There are many techniques to solve regression testing problems such as test case selection, test case prioritization or hybrid approach. Authors propose various strategies for test case prioritization. These include code coverage, non-code coverage and many other. Details of these techniques are given below.

A. Code Coverage based Test Case Prioritization

Rothermal et al. [3] investigate coverage based prioritization by examining a wide range of traditional prioritization techniques for specific objective function to give insight into trade off among these techniques for test case prioritization. They conduct their experiments for early rate of fault detection and measure efficiency of the approach by APFD matrix. The authors conclude that additional FEP prioritization is most suitable than all the other prioritization techniques that are based on coverage; however the total increase in APFD is not significant.

Li et al. [2] proposed a technique for prioritization of test cases for code coverage. They conduct an experiment to compare greedy, metaheuristics and evolutionary search algorithms to see the best algorithm for test case prioritization and explore factors that have significant importance in prioritization of test cases. They perform experiment on six programs; size and coverage is primary criterion. Results indicate that size of program does not but the size of the test suite directly affects prioritization complexity since it determines the size of the search space. Authors in [2] propose coverage based metrics proposed for test suite prioritization which gives high value of coverage effectiveness to those test cases which cover test requirements more quickly [2].

B. Non Coverage based Test Case Prioritization

Korel present model based test case prioritization and concluded that on average some model based tests prioritization methods might improve the effectiveness of early fault detection as compared to random prioritization [18]. In [19] for early rate of fault detection, author proposes an innovative equation for prioritization in time constraint environment. The authors validate their results through an experiment on eight C programs and on one case study and compared with random technique. They use APFD metric to measure detected faults and proved it as more effective in test case prioritization under time constraint.

C. AI Techniques for Test Case Prioritization

Artificial intelligence algorithms are widely used in software testing approaches [20, 21]. Walcott et.al present an approach for test case prioritization by using Genetic algorithm as a regression technique under time constrained which is based on coverage information (block and method). The authors compare effectiveness of genetic algorithm using APFD values with different ordering of test cases. The authors find it most effective in terms of rate of fault detection [4].

In contrast of this work Zhang et.al use ILP (integer linear programming) for test case selection and customary techniques for prioritization. The authors also compare traditional techniques, genetic based techniques and ILP. Their experimental results show that, ILP-based techniques are more effective over time than GA-based techniques [1].

Kaur et al [16] propose hybrid PSO algorithm for the prioritization of test case for regression testing in order to obtain maximum fault coverage in minimum execution time. They use PSO with GA to generate diversity in population and they also make use of APFD metric to asses' effectiveness of proposed algorithm finally showing its efficacy up to 75.6% for fault coverage.

III. Particle Swarm Optimization

PSO is a population based stochastic optimization technique proposed by Kennedy and Eberhart [5]. It is used to investigate the given search space to produce the optimal solution of declared problem. The search space comprises of 'n' particles and the collection of these particles is known as swarm. PSO searches for solution with the help of some parameters. Population of particles is initialized randomly and search for solution is done by updating particle's position and velocity. Each particle has memory to store its position pbest and best position among whole particles is known as gbest. Position is updated by adding velocity in previous position. Velocity is constrained by Vmax; to ensure that particles will search for optimal solution in defined search space. Velocity and position are updated using the following two equations (1) and (2).

$$Vik+1 = w * Vik + c1 * r1 * (SPBik - Sik) + c2 * r2 * (SGBik - Sik) \dots\dots\dots (1)$$

$$Sik+1 = Sik + (Vik+1) \dots\dots\dots (2)$$

where:

Vik :	velocity of particle i at iteration k
Sik:	current position of particle i at iteration k,
w:	inertia weight,
c1,c2:	constant weighting factors
SPBik:	local best of particle i at iteration k
SGBik:	global best of particle i at iteration k

Conclusively, in PSO each step is updated and validated in search of optimal solution. A particle is updated according to global and local best. We present Pseudo code of general PSO in Listing 1.

LISTING 1: Pseudo code for general PSO

```
Step 1
For
        Initialize Population Si where i=1,2,3 …… n
End
Step 2

For     Position of Particle Si,
        Calculate Fitness Value  Fi(k+1)
        If Fi(k+1) better than Fi(k)
        Set SPBik(Current Position) as the new Personal Best
        (Pbest) for the kth iteration
        Choose the particle with best Fitness Value as Global Best
        (SPGik) for kth iteration
        Calculate particle velocity Vik+1 from Eq (1)
        Calculate particle position Sik+1 from Eq (2)
End
```

IV. Proposed Approach

A. Proposed Factors

We propose an algorithm using the following factors.

Customer Priority: Customer priority is the importance of requirement to customer. Customer grades the specific

requirement by assigning value in the range from 1 to 10 according to significance of that requirement. The highest priority of the customer is denoted by10 [8, 10, 22].

Implementation complexity: Developers measure the implementation complexity of requirements by analyzing every single requirement in development effort point of view. Complexity is being rated from 1 to 10 [8, 23].

Requirement volatility: In literature, requirement volatility is taken as one of most important prioritization factor ranging from 1-10 [8, 9, 10].The volatility of requirements is keeping record of number of modifications of requirements from the time, the requirement was initially introduced.

Requirements traceability: it is the correlation of different software development artifacts such as software requirement specification and design document [14, 24]. It is proven that it should be an important factor to enhance quality of software [12].

Execution time: test case cost refers to operational time of test cases [8, 9, 13, 5]. Resource expenses are considered to be cost for software and execution time is considered to be one of these costs.

Fault impact of requirement: It is the identification of requirements that have more malfunctions in earlier version [9].The efficiency of test case can be improved by focusing on the functionalities that have greater number of failures [8, 11].

Our goal of prioritization is to increase the probability of revealing maximum faults earlier in testing process. Evolutionary algorithms can be used to solve test case prioritization problem. We have used modified version of PSO for earlier detection of faults and computed the results on three medium size projects.

B. Experimental Setup

We propose following steps to perform our experiments for validating our approach. These steps are given below:

1) Initial population is randomly generated (this is swarm of test cases in our case).

2) A particle is known as individual test case.

3) Particle's position represents the priority of the test case to be executed.

4) Standard equation of velocity is used to calculate particle's velocity.

5) Stopping criteria is needed to be fulfilled.

C. Proposed Algorithm

We show our proposed algorithm in Algorithm 1. We present this algorithm as steps. We solve prioritization problem by using equations of PSO algorithm with some alterations. The Position value of the test cases has to be integers in order to represent priority. The proposed algorithm uses fitness function values and the velocity equation values to calculate optimal order of the test cases.

Our proposed algorithm starts with generation of particle's population. We initially order their position (priority) sequence wise and calculated velocity on basis of particle's priority and original PSO technique respectively.

We calculate velocity of particle from its standard equation. We use rate of velocity to change the current positions of test case to the new position. We decide which particles have more fitness function values to execute earlier.

We explore relationship of velocity and fitness function in these calculations. We assign particles having lowest velocities more fitness value. We do not use standard equation of position for particles to reset their position, instead we use the knowledge of velocity to reset particle's position.

In other words, we assign priority on basis of velocity knowledge. Particle's position or its priority has clear cut importance in our approach. Algorithm suspends its execution on meeting stopping criteria. Our stopping criterion is dependent upon maximum number of iterations or full optimized solution that is when results will be constant.

Algorithm 1: Pseudo code for general PSO

Step I

Initialize No of Particles n = No. of Test cases (i=1,2,.....,n)

Step II

Set Position of Each Particle randomly Si

Step III

For k =1:p (Run a Loop for p Iterations)

{

Calculate ∑Ci (Value of Factors to be maximized obtained from position of ith particle)

Calculate ∑Ei (Value of Factors to be minimized obtained from position of ith particle)

Fi = (∑Ci-∑Ei)/ne (Fitness Fucntion)

where ne = no. of test cases executed

If Fi(k)>Fi(k-1)

SPBi = Si(k)

Else

SPBi = Si(k-1)

End

SGBi = Maximum (Fi) where i=1,2, ... N

Calculate Vik of each particle position

Update Positions Sik for each particle

}

END

Flow Chart

We present an overall diagram of our proposed approach mechanism in Fig. 1.

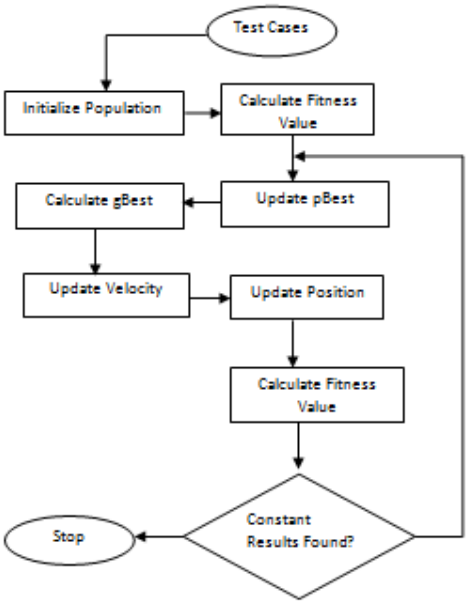

Fig. 1. Flow of proposed VBPSO algorithm

V. EVALUATION

We use three medium sized project in order to show the effectiveness of our proposed VBPSO algorithm. We present details of these projects in Table 1.

TABLE I. PROJECT DESCRIPTIONS

Attribute Description	Project 1	Project 2	Project 3
Project nature	Web based	Web based	desktop
No. Of functions	14	9	23
Test Cases length	40	21	47
Difficulty level	Medium	Medium	Medium
Team size	9	6	5

Customer was responsible to provide requirements and the priority of the each requirement. We involved project managers to give their expert opinion about ranking of requirements and to reduce the effect of biasness in rating process by using value based requirement prioritization tool [7]. We use Microsoft excel 2007 for requirement-test case traceability. We implement algorithm in MATLAB 9.0. and we list involved stakeholders in Table II below.

TABLE II. DATA SET

Factors	Values	Stakeholders
Customer priority	1- 10	Customers
Implementation complexity	1- 10	Developer
Requirement volatility	1- 10	Business Analyst
Requirement traceability	1- 10	Maintenance Engineer
Execution time	1- 10 sec	Developer
Fault impact of requirement	1-5	Test Engineer

We report 20 test cases in random order and we execute them in that order and subsequently run all test cases and detect faults. We then compute mean value of all results and subsequently use APFD metric to compare efficacy of proposed and random technique.

$$APFD = 1 - TF1 + TF2 + \cdots \ldots . . + TFM/NM + 1/2N \quad (4)$$

Where:

- T is the test suite under test.
- M is the number of faults in the program under test P.
- n is the total number of test cases.
- TFi is the position of the first test in T that reveals fault i.

We present list of parameters used in our proposed algorithm in Table III.

TABLE III. VBPSO PARAMETERS

Projects	Population size	Number of iterations	Termination criteria
Project 1	40	30	Constant results or iterations=30
Project 2	21	30	Constant results or iterations=30
Project 3	47	30	Constant results or iterations=30

VI. RESULTS & DISCUSSION

Other than APFD metric, results were also compared by analyzing percentage of executed test cases in finding of percentage of faults. This is important because regression testing often ends without performing all test cases. Considering the Project 1, we report that we obtain 42 % fault detection via PSO after executing 40% of test cases; and we detect 24% faults through random technique. We present our findings in Fig 2.

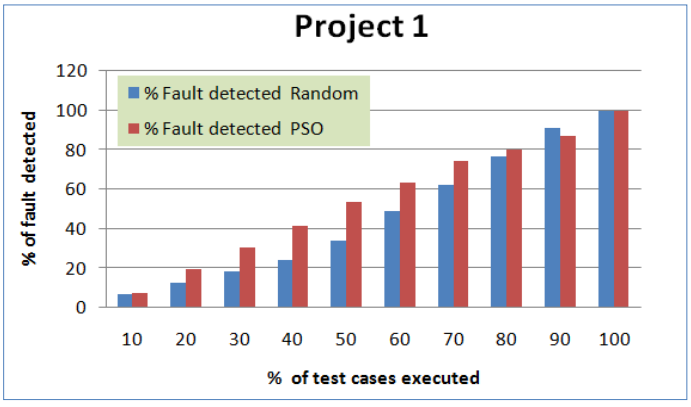

Fig. 2. Project 1 Results

We detect 20% of faults through random techniques and 39% through VBPSO in Project 2, as shown in Fig 3.

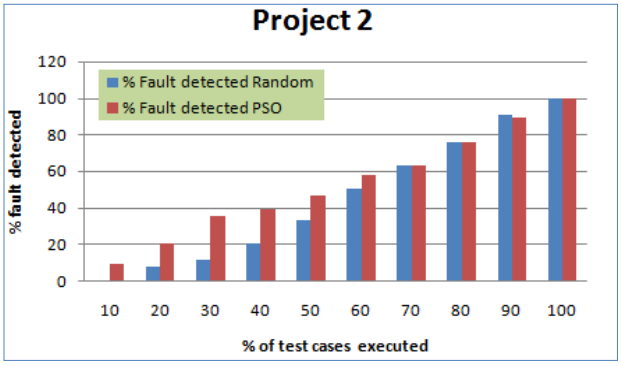

Fig. 3. Project 2 Results

We report this ratio as 45 % and 49 % through random and proposed algorithm respectively if we execute 40% test cases. We show our findings graphically in Fig 3. This shows a clear difference in detection of faults in case we cannot afford to execute whole test suite. As regression testing endures not only limited resources to perform but also gets higher expectation of maximum fault detection in earlier testing life cycle. So it is desired to perform it in a way to detect faults earlier. Our proposed algorithm resolves this problem. We can achieve higher earlier fault detection percentage while executing limited set of percentage of test cases.

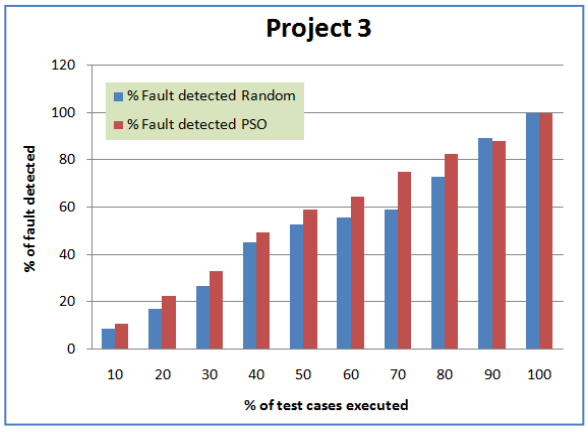

Fig. 4. Project 3 Results

We have also validated our results through APFD metric. In first project APFD calculation shows that VBPSO detects 78% faults whereas random ordering produces 67% of faults. In second project APFD rate through VBPSO was 67% while random ordering rate was 40%. In third project APFD results demonstrate that proposed algorithm detects 66% faults while random ordering produces 55% of faults that refers our algorithm as more effective.

We can have more refine APFD results if we slightly modify our algorithm's factor weight age in fitness function. We are accommodating six factors while analyzing earlier fault detection. But we can still work for it while considering subset of these factors such as execution time. We have found it very important to prioritize test cases in their true sense in order to deploy a quality and successful product. We present, in Table IV, tabular comparison of VBPSO and random fault detection for all these projects.

TABLE IV. APFD RESULTS

Approaches	P1	P2	P3
Random	67%	40%	55%
VBPSO	78%	67%	66%

Experimental results show that the proposed algorithm was able to detect more fault than random technique. Furthermore it is depicted by fault detection rate that, there is still room for improvement. However, achieving such a high fault detection rate proves the competiveness of our technique as compared to other existing approaches.

VII. CONCLUSION & FUTURE WORK

We present a hybrid approach of artificial intelligence with value based concept to solve prioritization problem in regression testing. Concept of value has been used to involve stakeholder's participation in process via proposing a set of six different factors. Our analysis shows the percentage of faults detected in prioritized test suite with the help of APFD.

Our results show the effectiveness of our proposal by evaluating three medium sized projects. We prove an overall effectiveness of our proposal for early fault detection.

REFERENCES

[1] Lu Zhang, Shan-Shan Hou, Chao Guo, Tao Xie, Hong Mei "Time Aware Test-Case Prioritization using Integer Linear Programming",ISSTA'09, July 19–23, 2009, Chicago, Illinois, USA

[2] Z. Li, M. Harman, and R.M.Hierons "Search Algorithms for Regression Test Case Prioritization",IEEE Transaction on Software Engineering, VOL. 33, NO. 4, APRIL 2007

[3] G. Rothermel, R. Untch, C. Chu and M. Harrold, "Test Case Prioritization: An Empirical Study" International Conference on Software Maintenance, Oxford, UK, pp. 179 - 188, September 1999

[4] Kristen R. Walcott. Mary Lou Soffa "Time Aware Test Suite Prioritization", ISSTA'06, July 17–20, 2006, Portland, Maine, USA

[5] Khin Haymer Saw Hla, YoungSik Choi, Jong Sou Park "Applying Particle Swarm Optimization to Prioritizing Test Cases for Embedded Real Time Software Retesting", 8th International Conference on Computer and Information Technology Workshops IEEE 2008

[6] J. C. Munson and S. Elbaum, "Software reliability as a function of user execution patterns and practice," 32nd Annual Hawaii International Conference of System Sciences, Maui, HI, pp. 255-285, 1999

[7] B. Boehm, "Value-Based Software Engineering," ACM Software Engineering Notes, vol. 28, pp. 1-12, March 2003.

[8] R. Krishnamoorthi, S.A. Sahaaya and Arul Mary "Incorporating varying Requirement Priorities and Costs in Test Case Prioritization for New and Regression testing", 2008

[9] X. Zhang, C.Nie, B. Xu and B.Qu "Test Case Prioritization based on Varying Testing Requirement Priorities and Test Case Costs", 2007

[10] H. Srikanth, L. Williams and J. Osborne "System Test Case Prioritization of New and Regression Test Cases", 2005

[11] T. Ostrand, E. Weyuker and R. Bell, "Where the Bugs Are," Proceedings of the ACM SIGSOFT International Symposium on Software Testing and Analysis, Boston, MA, pp. 86-96, July 2004

[12] A. Ahmed, "Software Testing as a Service" Auerbach Publications, New York: 2009

[13] A. M. Smith, G. M. Kapfhammer "An Empirical Study of Incorporating Cost into Test Suite Reduction and Prioritization", 2009

[14] R. Krishnamoorthi and S.A. Mary "Factor oriented requirement coverage based system test case prioritization of new and regression test cases",2009

[15] "value." Merriam-Webster Online Dictionary. 2008. Merriam-Webster Online. 23 October 2008, http://www.merriam-webster.com/dictionary/value

[16] A.Kaur and B.bhatt "Hybrid Particle Swarm Optimization for Regression Testing"International Journal on Computer Science and Engineering (IJCSE) Vol. 3 No. 5 May 2011

[17] B. Boehm, "Value-Based Software Engineering", ACM SIGSOFT, March 2003.

[18] B. Korel "Application of System Models in Regression Test Suite Prioritization" 2008

[19] Y. Fazlalizadeh, A. Khalilian, M. AbdollahiAzgomi and S. Parsa "Prioritizing Test Cases for Resource Constraint Environments Using Historical Test Case Performance Data" IEEE 2009

[20] Kumar, Sushant, PrabhatRanjan, and R. Rajesh. "Modified ACO to maintain diversity in regression test optimization." *2016 3rd International Conference on Recent Advances in Information Technology (RAIT)*.IEEE, 2016.

[21] Solanki, Kamna, et al. "Test Case Prioritization: An Approach Based on Modified Ant Colony Optimization." *Emerging Research in Computing, Information, Communication and Applications*. Springer Singapore, 2016. 213-223.

[22] Nayak, Soumen, Chiranjeev Kumar, and SachinTripathi. "Effectiveness of prioritization of test cases based on Faults." *2016 3rd International Conference on Recent Advances in Information Technology (RAIT)*.IEEE, 2016.

[23] Muthusamy, Thillaikarasi. "A Test Case Prioritization Method with Weight Factors in Regression Testing Based on Measurement Metrics." *International Journal* 3.12 (2013).

[24] Thillaikarasi Muthusamy and K. Seetharaman, "Efficiency of Test Case Prioritization Technique Based on Practical Priority Factors". *International Journal of Soft Computing,* 10 (2015) 183-188.