International Journal of Current Trends in Engineering & Technology
ISSN: 2395-3152
Volume: 03, Issue: 01 (JANUARY– FEBRUARY, 2017)

Test-Case Prioritization Using Binary Particle Swarm Optimization Method

Shruti Mishra
Computer Science & Engineering SSSIST, Sehore
Bhopal, India
shruti.mishra.766@gmail.com

Kailash Patidar
Computer Science & Engineering SSSIST, Sehore
Bhopal, India
kailashpatidar123@gmail.com

Abstract- Particle swarm optimization method is based on artificial intelligence technique. It is an optimization method that was developed in 1995 by Eberhart and Kennedy based on the social behaviors of fish schooling or birds flocking. By increasing the overall rate of fault detection, a greater number of errors can be found more rapidly in the code. Particle , fitness function , local best , global best , velocity update , position update are the commonly used elements in particle swarm optimization.PSO algorithms have been developed to solve constrained problems, multi-objective optimization problems, problems with dynamically changing landscapes, and to find multiple solutions. On the other hand some of them defined different methods like inertia weight to improve the performance of PSO.

Keywords- Optimization, Prioritization, artificial intelligence, Swarm, inertia, constriction.

I. INTRODUCTION

Software testing is critical to software production. With day by day increase in software system complexity and competition in business, effective software test methods and automation tools are strongly needed in the real world in order to deliver high quality software products in product schedules to back support engineers. Software testing is the process of experimenting a program with well-designed input data (called test-cases) to capture failures. More explicitly saying software testing is the process of executing a program with the intent of locating errors to prevent failures. Testing identifies faults and by removing faults we can increase the quality of software. Testing also measures the software quality in terms of its capability for achieving accurateness, reliability, usability, maintainability, reusability, correctness and testability. The objective of testing varies according to the problem, process and level of testing. In general we can say that the objectives of testing are: Executing a program with prime goal of finding an error. A good test is having a high probability of finding an as-yet-undiscovered error. To design test case so that it can find the error in minimum amount of time and effort.

1.1 Test Case Prioritization

Test case prioritization techniques [12] [13] provide an ordered sequence of test cases for execution on the basis of following testing goal or objective that maximization of fault detection capability and code coverage capability of test suites [13] Model based testing refers to software testing where test cases are derived in whole or in part from a model that describes some (usually functional) aspects of the system under test (SUT).

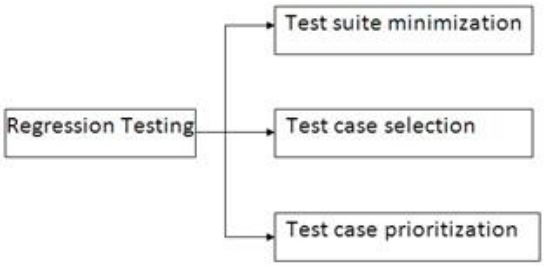


Figure 1 Regression Testing

The basic idea of model-based software testing is to identify and build abstract model(s) to present certain properties and behaviors of the under-test software product so that different kinds of model-based testing activities can be performed efficiently as follows: Efficient and systematic model-based test planning (test modeling and analysis). Systematic model-based test design, generation, execution, and result validation. Effective test suit updates and reuse using model-based approaches. Accurate model based test coverage analysis and quality evaluation.

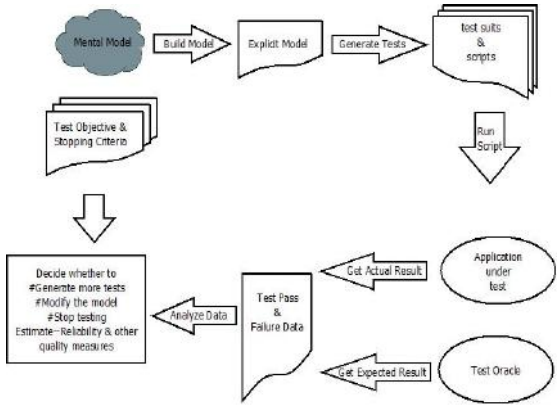


Figure 2 Activities in Model Based Testing

1.2 Test Case Optimization

We call a test case is good, if it cover more features of test objective and eliminating redundant test cases. In other word we can say that testing process relies on the quality of test cases not in quantity of test cases for better results. By eliminating of redundant test cases it will save time. Therefore, automatic generation of test cases has reduces some work load from the tester and the developer, and it also saves cost and time. [12]Several tools have been proposed to automate or optimize software testing tasks, from test generation to its execution. Regarding automatic Test Case generation, we can identify tools which generate test suites from some software artifact (such as code, functional requirements, and use cases). However, as these tools generate Test Cases in a systematic way (aim into provide a good coverage of the testing adequacy criterion), the generated test suites are usually too large to be executed with the available resources (tools, time, people). When analyzing large test suits, we can identify

redundancies in Test Cases. Hence, it is possible to cut down these suits, in order to fit the available resources, without severely compromising the coverage of the test adequacy criterion being observed. The task of reducing a test suite based on some selection criterion is known as "Test Case selection". The test selection criterion depends on the test adequacy criterion being used. Clearly, Test Case selection should not be performed at random, in order to preserve the coverage of the testing criterion. In the absence of automatic tools, this task is usually manually performed in an ad-hoc fashion. However, manual Test Case selection is time-consuming and susceptible to errors. Different authors try to automatically solve the Test Case selection problem by deploying a variety of techniques. Some works focus on deterministic software engineering solutions. Despite their good results, these works consider only a single criterion for test case selection.

II. Literature Survey

[1]Particle Swarm Optimization is a biologically inspired computational search and optimization method developed in 1995 by Eberhart and Kennedy based on the social behaviors of birds flocking or fish schooling. A number of basic variations have been developed due to improve speed of convergence and quality of solution found by the PSO. On the other hand, basic PSO is more appropriate to process static, simple optimization problem. Modification PSO is developed for solving the basic PSO problem. The observation and review 46 related studies in the period between 2002 and 2010 focusing on function of PSO, advantages and disadvantages of PSO, the basic variant of PSO, Modification of PSO and applications that have implemented using PSO. The application can show which one the modified or variant PSO that haven't been made and which one the modified or variant PSO that will be developed. [2]Test case prioritization techniques schedule test cases in an order that increases their effectiveness in meeting some specific goal. One performance goal, rate of fault detection, is a measure of how quickly faults are detected within the testing process; an improved rate of fault detection can provide faster feedback on the system under test, and let software engineers begin locating and correcting faults earlier than might otherwise be possible. In previous work, we reported the results of studies that showed that prioritization techniques can significantly improve rate of fault detection. Those studies, however, raised several additional questions:(1) can prioritization techniques be effective when aimed at specific modified versions (2) what tradeoffs exist between fine granularity and coarse granularity prioritization techniques (3) can the incorporation of measures of fault proneness into prioritization techniques improve their effectiveness. [3] Particle Swarm Optimization (PSO) is a relatively recent heuristic search method whose mechanics are inspired by the swarming or collaborative behavior of biological populations. These two evolutionary heuristics are population-based search methods. In other words, PSO and the GA move from a set of points (population) to another set of points in a single iteration with likely improvement using a combination of deterministic and probabilistic rules. The GA and its many versions have been popular in academia and in the industry mainly because of its ease of implementation, and the ability to effectively solve highly nonlinear, mixed integer optimization problems that are typical of complex engineering systems. The only drawback of the GA is its expensive computational cost. This paper attempts to examine the claim that PSO has the same effectiveness (finding the true global optimal solution) as the GA but with significantly better computational efficiency (less function evaluations) by implementing statistical analysis and formal hypothesis testing. The performance comparison of the GA and PSO is implemented using a set of benchmark test problems as well as two space systems design optimization problems, namely, telescope array configuration and spacecraft reliability-based design. [4] The time taken performing the fitness calculations can dominate the total computational time when applying to Particle Swarm Optimization (PSO) to complex real life problems. This paper describes a method of estimating fitness, and the reliability of that estimation, that can be used as an alternative for performing some true fitness calculations. The fitness estimation is always made but should the reliability of this fitness estimation drop below a user specified threshold, the estimate is discarded and a true fitness evaluation performed. Results are presented for three problems that show that the number of true fitness evaluations can be significantly reduced by this method without degrading the performance of PSO. Further the value used for the threshold, the only new parameter introduced, is shown not to be sensitive, at least on these test problems. Provided that the time to perform a true fitness evaluation is far longer than the time for the fitness and reliability calculations, a substantial amount of computing time can be saved while still achieving the same end result. A given particle has both a fitness (either evaluated or estimated) and a reliability that gives an indication of how reliable that fitness is thought to be. A fitness that is truly evaluated has a reliability of unity, but with each fitness estimation the reliability of the estimated fitness reduces. When estimating the fitness at some point, should the reliability of this estimate drop below a user specified threshold the estimate is abandoned and a true fitness evaluation is made(thus restoring the reliability to unity). To minimize the number of positions whose fitness and reliability need to be kept, only the positions occupied by the particles in the previous iteration are kept. The fitness of a particle after it has been moved is derived from the fitness and reliability of this particle before it was moved and the fitness and reliability values associated with the position of the particle that was closest to this new position last iteration. [5] Particle swarm optimization is a heuristic global optimization method and also an optimization algorithm, which is based on swarm intelligence. It comes from the research on the bird and fish flock movement behavior. The algorithm is widely used and rapidly developed for its easy implementation and few particles required to be tuned. The main idea of the principle of PSO is presented; the advantages and the shortcomings are summarized. At last this paper presents some kinds of improved versions of PSO and research situation, and the future research issues are also given. In the basic particle swarm optimization algorithm, particle

swarm consists of "n" particles, and the position of each particle stands for the potential solution in D-dimensional space. The particles change its condition according to the following three principles: (1) to keep its inertia (2) to change the condition according to its most optimist position (3) to change the condition according to the swarm's most optimist position. The position of each particle in the swarm is affected both by the most optimist position during its movement (individual experience) and the position of the most optimist particle in its surrounding (near experience). When the whole particle swarm is surrounding the particle, the most optimist position of the surrounding is equal to the one of the whole most optimist particle; this algorithm is called the whole PSO. If the narrow surrounding is used in the algorithm, this algorithm is called the partial PSO. Each particle can be shown by its current speed and position, the most optimist position of each individual and the most optimist position of the surrounding. In the partial PSO, the speed and position of each particle change according the following equality. [6] Genetic Algorithm (GA) is heuristic search algorithm. They are based on the idea of natural selection and genetics. This algorithm is inspired by the Dalton's theory about evolution that is survival of the fittest. It is a part of evolutionary computing which is a growing field of Artificial Intelligence. GA exploits the historical information to direct the search in region of better performance with in the search space. In this way competition among individuals for better resources results in fittest individual dominating over the weaker ones. Genetic Algorithms are robust hence they are better than conventional AI (Artificial Intelligence). GA does not break easily even if there is a slightly change in the input and it also does not get affected by noise. GA offer benefit over typical search of optimization problem in searching large state space, multi modal search space and n dimensional surface. GA consists of following operators that is Selection Operator chromosomes are selected for cross-over based on the value of the fitness function Cross- Over Operator combine two chromosomes to produce new chromosomes. Mutation Operator in this value is randomly changed to create new genes in the individual. GA starts with randomly generated population of individuals or chromosomes .Fitness of individual is calculated based on some fitness function. After the fitness is calculated selection of individuals is done based on the Roulette -Wheel Selection method i.e an individual having higher fitness value has the more chance of getting selected .Then Cross over operator is applied to produce new offspring in the population that may have better characteristics than their parents. Mutation is done to introduce new individual in the population is done by flipping the bit of the chromosomes. Particle Swarm Optimization is a relatively recent heuristic search method. It is similar to GA in the sense that both are evolutionary algorithms .It is one of the meta-heuristics approach that optimizes a problem and try to improve candidate solution iteratively. PSO is generally used to solve those problems whose solution can be represented as a point in an n-dimensional space. In PSO potential solution is called particle. A number of particles are randomly set into motion through this space. Each particle posses its current position, current velocity, and its pbest position. Pbest is the personal best position explored so far. It also incorporates Gbest that global best position achieved by all its individuals. It is a simple approach and it is effective across a variety of problem domains. Pseudo code for PSO the PSO algorithm consists of just few steps, which are repeated until some stopping condition is met. The steps are as follow: Initialize the population of individuals with current position and, velocity. Evaluate the fitness of the individual particle (P best). Keep track of the individual highest fitness (G best). Modify velocity based on Pbest and Gbest location. Update the particle position. PSO starts with initialization of particle velocity and current position. Here particle is in 2-D space . Fitness value of the particle is calculated according to function. If the fitness of the particle is better than its previous value update particle x and y position that is its personal best position. Also if the value is better than gbest position update global best position of the particle. Apply equations to update the x and y velocity vector of the particles. Process repeats until termination criteria are met or the optimal solution is found.

III. Proposed Solution

Here the particle swarm optimization concept is used which consists of updating the velocity of accelerating each particle toward its p*best* and *gbest* locations at each time step. Acceleration is weighted by a random term, with separate random numbers being generated for acceleration toward *pbest* and *gbest* locations.

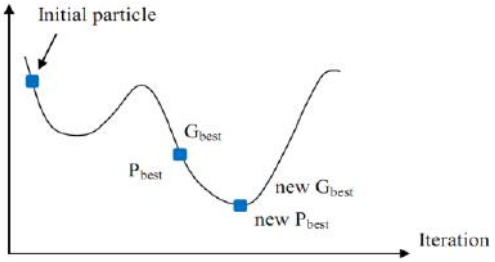


Figure 3 Gbest and Pbest converge curve

Let f be the objective function to be maximized, then, the personal best position of a particle at iteration or time step t is updated as follows:

$$y_i(t) = \begin{cases} y_i(t-1) & \text{if } f(x_i(t)) \le f(y_i(t-1)) \\ x_i(t) & \text{if } f(x_i(t)) > f(y_i(t-1)) \end{cases}$$

Where y_i (t) is personal best position and x_i (t) is swarm current position. For the gbest model, the global best position is determined from the entire swarm by selecting the best personal best position. This position is denoted by. The equation that manipulates the velocity is called the velocity update equation and is stated as follows:

$$V_i^{k+1} = wv_i + c_1 r_1 (pbest_i - s_i) + c_2 r_2 (gbest_i - s_i)$$

Where v_i^{k+1} is the velocity updated for the *j*th dimension. C_1 is the acceleration constants where moderates the maximum step size towards the personal best position of the particle. C_2 is the acceleration constants where moderates the maximum step size towards the global best position in iteration. r_1 (t), r_2 (t)

is two random values in the range [0, 1] which give the PSO algorithm a stochastic search property. w is Inertia weight.

Inertia component, W–which serves as a memory of the previous flight direction for example movement in the immediate past.

Cognitive component, C1– which quantifies the performance of particle i relative to past performances, which means the cognitive component, resembles individual memory of the position that was best for the particle.

Social component, c_2 quantifies the performance of particle i relative to a group of particles. The effect of the social component is that each particle is also drawn towards the best position found by the particle's neighborhood. Velocity updates to each dimension can be clamped with a user defined maximum velocity, Vmax, which would prevent them from exploding, thereby causing premature convergence. Each particle updates its position using the following equation:

$$s_i^{k+1} = s_i^k + v_i^{k+1}$$

Binary Particle Swarm Optimization

In this each particle uses binary values to represent its current position and the position of the best solution found. The velocity vector is updated as in the continuous version, but determining the probability that each bit of the position vector becomes 1. The velocity vector should be mapped in such a way that it only contains values within the [0, 1] range. To this end, the sigmoid function is applied to each of its values. The equation for sigmoid function and updating positions is then replaced by the following probabilistic update equation:

$$x_{ij} = \begin{cases} 1 & \text{if } r < s(v_{ij}) \\ 0 & \text{otherwise} \end{cases} \quad \text{with} \quad s(x_{ij}) = \frac{1}{1+\exp(-x_{ij})}$$

where s (t) is a random value in the range [0, 1]

Step 1: Making CFG: Convert the activity diagram into a directed graph as Control Flow Graph (CFG). Elements showing an activity of activity diagram are treated as nodes in CFG. Flow is also shown through directed graph between the nodes. If the edge is outward from the node that means it is calling the other. If the edge is inward to the node that means it is being called by the other. Node with no inward edge is initial node and that of with no outward edge is final node.

Step 2: Assigning Weights: Based on the Information of Flow metrics (IF) concept every node is designated with a weight value. This IF value is defined as: IF (X) = FAN IN (X) * FAN OUT (X) where, FAN IN (X) is defined as number of inward edges to node X. FAN OUT (X) is defined as a number of outward edges to node X.

Step 3: From source to destination there are many paths possible by traversing the CFG. Complexity of a path may be calculated by using basic IF model i.e. summing of all the weights corresponding to every node in a path selected. The Fitness formula gives fitness value (F) of path P and it can be calculated on the basis of following formula:

$$F_{path=P} = \sum_{i=0}^{n} W_i$$

where, W_i is weight of i^{th} node in P^{th} path

Step 4: Identification of decision node is done and collectively they form a test data. We call it chromo-particle (as like called chromosome in [22]). Chromo-particle is a binary string or an individual in the population and every bit is corresponding to a decision node. Every chromo-particle refers to a unique path (generated by traversing source to destination). The fitness value of this path can be calculated by equation 3.2. Any change in bit values of this chromo-particle generates a new path. Hence our target is to generate such chromo-particle which can give higher fitness value. And for this Binary PSO is used to get the optimum chromo-particle. Note: Chromo-particle with high fitness value is considered as a good solution. It is being presumed that no loop is taken more than once as also in [22].

Step 5: Finally binary particle swarm optimization algorithm is applied to this chromo-particle by initializing the velocity and particle position values. The algorithm is run until it finally reaches to some predefined number of iteration or any predefined fitness value. The Binary PSO finds the global best solution. This solution gives the highest fitness value so far found in iteration. And we are looking for the same higher fitness value. Better the fitness value higher the priority should be given to that test data. Binary particle swarm optimization algorithm is applied to chromo particle on some random population by initializing the particle and velocity values. The algorithm is run until it reaches to some predefined value of iteration or any predefined fitness value is obtained.

Proposed Methodology

Algorithm 1: Making CFG, Input: Activity Diagram Output: Weighted Control Flow Graph

1. Convert the activity diagram into control flow graph (CFG).
2. For every node in CFG assign weights by applying IF metric concept which outputs a weighted CFG.
3. Identify the decision nodes and which forms collectively a chromo particle.

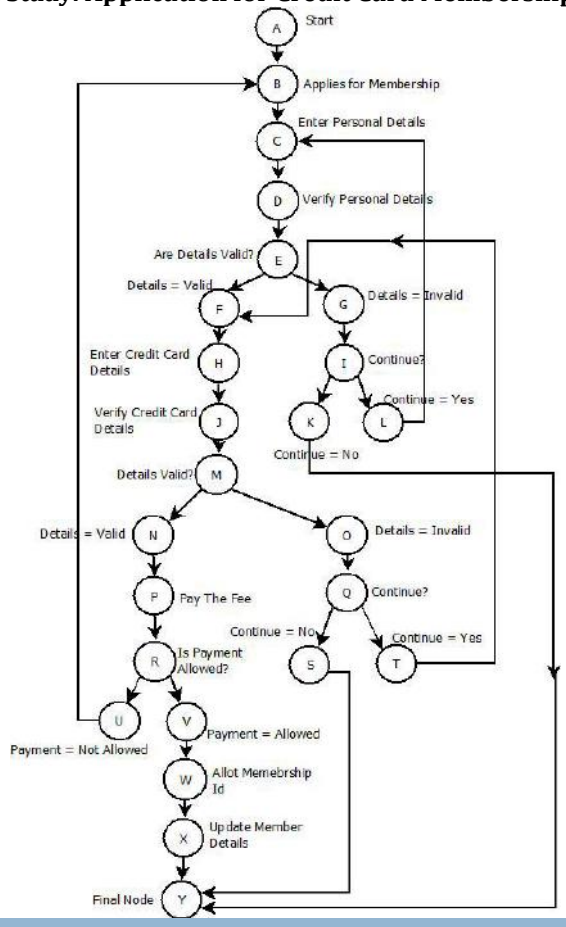
Proposed Algorithm:

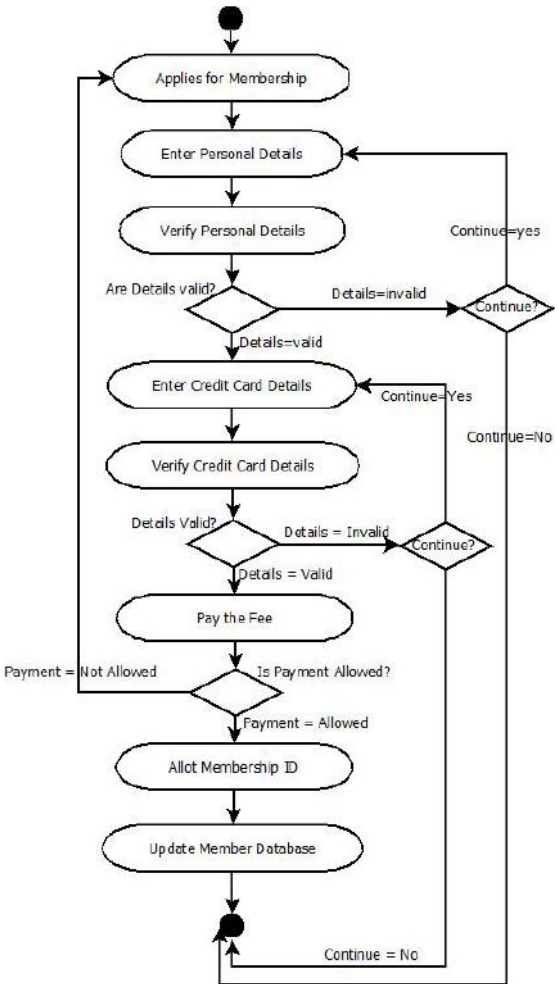
1. Input: Weighted Control Flow Graph Output: Highest Priority Test Case
2. Generate (or update) the test data chromo-particle population and its velocity using BPSO equations (initially randomized)
3. For x = 1...n
4. Depth first search (DFS) is applied to CFG to identify the paths.
5. fitness value for each test data is calculated by using fitness function (equation 3.2) on the corresponding path.
6. Local Best is calculated.
7. Calculate the Global Best value for all set of values.
8. If test data for all the values have not been covered, then repeat the BPSO process.

International Journal of Current Trends in Engineering & Technology
ISSN: 2395-3152
Volume: 03, Issue: 01 (JANUARY– FEBRUARY, 2017)

9. Else return the path corresponding to the last global value obtained.

Case Study: Application for Credit Card Membership





First Iteration

S.No	Particle	E	I	M	Q	R	Fitness_I
1	00001	0.357	0.315	0.759	0.033	0.321	20
2	11010	0.276	0.701	0.911	0.046	0.719	38
3	11110	0.778	0.683	0.796	0.292	0.125	47
4	10111	0.524	0.339	0.534	0.928	0.952	11

Second Iteration

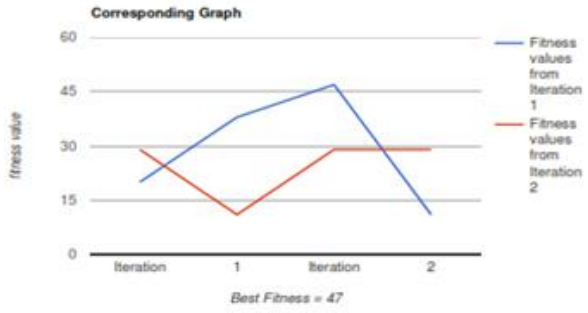
S. No	Particle	E	I	M	Q	R	Fitness_II
1	00000	0.656	0.039	0.564	0.997	0.249	29
2	10100	0.578	0.755	0.565	0.548	0.775	11
3	00000	0.914	0.680	0.272	0.316	0.040	29
4	00010	0.351	0.798	0.227	0.666	0.300	29

S. No.	Fitness_I	Fitness_II	Local Best, Particle
1	20	29	29, 00000
2	38	11	38, 11010
3	47	29	47, 11110
4	11	29	29, 00010

IV. Results and Analysis

Weight on CFG							
NODE	FI	FO	FI*FO	NODE	FI	FO	FI*FO
A	0	1	0	N	1	1	1
B	2	1	2	O	1	1	1
C	2	1	2	P	1	1	1
D	1	1	1	Q	1	2	2
E	1	2	2	R	1	2	2
F	1	1	1	S	1	1	1
G	1	1	1	T	1	1	1
H	2	1	2	U	1	1	1
I	1	2	2	V	1	1	1
J	1	1	1	W	1	1	1
K	1	1	1	X	1	1	1
L	1	1	1	Y	3	0	0
M	1	2	2				
Weight Assignment (FAN_IN = FI; FAN_OUT = FO)							

Local Best = Larger (Fitness I & Fitness II) Global Best = Larger of all Local Best Values. Hence Global Best Value so far from Iteration 1 & Iteration 2 is 47 and the particle is 11010



Conclusion

Binary Particle Swarm Optimization (BPSO) technique is used for identifying the best test path that must be tested first. The activity diagram is converted into control flow graph and further IF model is used to obtain the fitness function which is required in BPSO to calculate the best path. Previously a work is done in the same area using Genetic Algorithm but proposed approach has given the better result given in less number of iterations. There are several basic variant of PSO. The basic variants have supported in controlling the velocity and the stable convergence. Modified variant PSO help the PSO to process other conditions that cannot be solved by the basic PSO.

References

[1]. Dian Palupi Rini & Siti Mariyam Shamsuddin- International Journal of Computer Applications, January 2011.

[2]. Sebastian Elbaum & Gregg Rothermel- Technical Report # TR-UNL-CSE-2000 0005, August 2000.

[3]. R. Krishnamoorthi and S. A. Sahaaya Arul Mary- International Journal of Hybrid Information Technology, Vol.2, No.3, July, 2009.

[4]. Rania, Hassan Babak, Cohanim Olivier de Weck- Massachusetts Institute of Technology, Cambridge, MA, 02139.

[5]. Tim Hendtlass, Centre for Information Technology Research Swinburne University of Technology-IEEE 2007.

[6]. Qinghai Bai College of Computer Science and Technology Inner Mongolia University for Nationalities Tongliao 028043, China- Febuary 2010.

[7]. Application of Genetic Algorithm and Particle Swarm Optimization in Software Testing - IOSR Journal of Computer Engineering (IOSR-JCE), Deepti Arora, Anurag Singh Baghel, Volume 17, Issue 1, Ver. II (Jan - Feb. 2015)

[8]. Prof. Abdel-Hadi Nabih Ahmed & Prof. Atef M. A-Moneim - International Journal of Hybrid Information Technology, Vol.1, No.3, July, 2007.

[9]. S. Mirjalili, S.Z. Mohd Hashim, G. Taherzadeh, S.Z. Mirjalili, and S. Salehi - Faculty of Computer Science and Information Systems, University Teknologi Malaysia, 81300 Skudai, Johor Bahru, Malaysia - International Conference on Genetic & Evolutionary Method-2011.

[10]. Thillaikarasi Muthusamy & Seetharaman. K , "A New Effective Test Case Prioritization for Regression Testing based on Prioritization Algorithm" , International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868, Volume 6– No. 7, January 2014.

[11]. Ya-Hui Jia & Wei-Neng Chen, "Generating Software Test Data by Particle Swarm Optimization", School of Information Science and Technology, Sun Yatsen University, Guangzhou, China.

[12]. I. K. El-Far and J.A. Whittaker, "Model-Based Software Testing", John J.Marciniak, "Encyclopedia of Software Engineering", vol. 1, pp. 825837, Wiley-Inter Science, 2002.

[13]. L. Apfelbaum and J. Doyle, "Model-Based Testing", Software Quality Week Conference, May 1997.