Particle Swarm Optimization with Cross-Over Operator for Prioritization in Regression Testing

Dr. Arvinder Kaur
University School of Information Technology,
Guru Gobind Singh Indraprastha University

Divya Bhatt
University School of Information Technology,
Guru Gobind Singh Indraprastha University

ABSTRACT

Software Testing is continuous process of development and maintenance in life of software. In maintenance phase, regression testing gets exercisedwith additional resources/time for performance. The prioritization of test cases helps to reduce the cost-time of regression testing. Hence, completing Regression Testing effectively and on schedule is challenge for software tester. In this research paper, the Particle Swarm Optimization (PSO) technology has been studied and used with the blend of Genetic Algorithm (GA) and the hybrid prioritized algorithm has been proposed. The Particle Swarm Optimization is an optimization algorithm based on heuristic search which can be used to solve time-constraint environment of Test Case Prioritization and the concept of Genetic Algorithm will further help in diversifying the solution within whole search space. For finding the effectiveness of hybrid prioritization algorithm: the efficiency %, saving %, reduction % and APFD/APCC has been calculated.

KeywordsRegression Testing, Particle Swarm Optimization, Genetic Algorithms

1. INTRODUCTION

During maintenance phase, modifications/defects in software are corrected. Hence, it is very difficult to decide which test case should be executed, or which test case to mark as "effective" in order to detect the cause of modification in software. Regression Testing is done within maintenance phase. Regression Testing assures that modification done in software has not over-ruled user specification and expectations. Hence, to make Regression Testing more scheduled many criteria's can be followed: reduction of test case, test case selection, prioritization of test cases. In this research paper, the concentration is in the area of "prioritization of test cases". The time constrained prioritization of test cases can be reduced to 0/1 knapsack problem [1] which is a NP-hard [2] and solved using Particle Swarm Optimization [3] and Genetic Algorithm [4] techniques. Thus, combination of two optimization techniques has been used here, to make effective solution on-schedule.

The techniques described are: Particle Swarm Optimization (PSO) and Genetic Algorithm (GA) operator. The PSO is a global optimization algorithm based on heuristic search [5]. The idea was given by John Kennedy and Eberhert, in 1995 [6], after observing the group of animals, flock of birds and fishes, where each individual follows the path of "global best" particle with in its population, e.g. in ocean, while searching for good food source (plankton), a school of fish, travels everywhere making cylindrical shape. The inside story: every fish is observing its neighbor's position and velocity then compare it with global best position and velocity. The "best" position and velocity is chosen and updates are made by individuals in their position and velocity. Thence, each fish converges towards the best position after modification in its velocity, which helps to move towards the food faster.

In this paper, we have blended the PSO with crossover operator of GA [5], to avoid group of population from converging to local best. The GA process consists of [7]: selection, crossover and mutation. In GA, from general population, a set of best fitted population is selected on the basis of fitness function and crossover, mutation operators are applied to get optimal solution. The crossover operator helps in PSO to make each individual a widened look of search space, by crossing over them with other individuals. This resulted in hybrid prioritization algorithm of PSO & GA.

The PSO has been applied to NP-hard combinatorial problems Traveling Salesman Problem [8], and NP Knapsack [3] whereas GA is known for optimizing scheduling problems [9]. Here, we have tried to use combination of PSO and GA to solve, time-constrained prioritization and code based prioritization. To bring clarity between prioritized and non-prioritized test-suite Average percentage fault detection (APFD) [10] and Average percentage Condition Coverage (APCC) [11] has been calculated and represented using graphs. The APFD has been used to evaluate average percentage of faults detected as per average percentage of test-suites executed whereas APCC has been evaluated average number of condition covered as per average percentage of test suite executed.

2. RELATED WORK

The process of Regression testing is quite expensive and time consuming even for small systems. To address problem of regression testing many test case prioritization techniques have been explored by researchers such as: hybrid approach by Wong [12], Test Selection Algorithm by Aggarwal [13], version specific technique [14] where test cases are prioritized for specified version of software, hybrid technique based on variable method for regression testing [15], optimization algorithm techniques [16] by observing natural behavior of ants and swarms [17] for test case prioritization, code-coverage based technique [18] where internal structure is analyzed and therevalidation strategy [19] based on extension of Fischer

algorithm decides which Test case to re-run for Regression Testing, CBSE (component based software engineering) for regression testing where features associated with test cases are exercised [20], SBSE (search based software engineering) for regression test optimization [21, 22] MORTO (multi objective regression testing optimization) for selection and prioritization usage in real world [23] . The regression testing techniques use greedy approach [24] taking feedback from previous selection is fast to achieve.

Many researchers have explored area of PSO and on being an optimization approach it has been used to solve routing optimization [25] problems where the optimized path for traffic routing, packet routing is search. The scheduling problems such as: job-scheduling [26], task scheduling problems in distributed systems [27] where task or job is divided into subsets to make performance cost efficient. The PSO has been applied in the field of cryptography and crypt-analysis [28] for military applications such as code encryption and effective code security. The traveling salesman problems is to find optimal path to reach from source to destination with less time and cost, that make it NP-hard combinatorial problem has been solved using PSO technique [29] and other combinatorial problems such as: packing and knapsack [30]. The applications of detection of fault and recovering from them includes field: test pattern generation for circuits [31], software faults detection [32] have been applied by PSO technique.

The GA is favorable field of many scientists, explorers and researchers; this makes application of GA in the diverse fields. The field of regression testing and its techniques has been explored using GA and tools are available to effectively implement regression testing [33]. The GA allows data to evolve naturally and it makes logical patterns that help in information processing for trend, financial analysis such as: data mining [34]. The GA is efficiently been used in the field ofencryption and code breaking [35]. The GA approach can be used for pattern detection for the learning accuracy improvement and noise filtering [36] of image.

3. HYBRID PRIORITIZED ALGORITHM

In PSO, while searching for best solution in the search space, the particle can converge towards point between particle best and local best (neighbor best), this results particle to converge towards single point, ignoring other aspects of search spaces (ignoring global best conditions).To overcome this limitation, GA factor has been used along with PSO algorithm to propose hybrid prioritized algorithm. The reason to apply crossover [5] in PSO is to increase population diversity and ability of PSO to avoid local maxima [37] and make searching process fast.

3.1 Assumptions

a) **Test cases** of given problem is equal to particles in population.
b) Randomly generate initial population.
c) The position, velocity and stopping criteria of particles depend on problem.
d) In our case, two lists are maintained:
   (1) Position= rank of test case based on faults or conditional node covered.
   (2) Velocity= execution time or independent paths covered.
e) **Stopping Criteria** is total number of faults covered in minimum time and independent path covered.

Input: 'n' number of Test Case is selected.

Output: Test suite with minimum number of test cases and full-filling stopping criteria.

3.2 Algorithm

Proposed Algorithm
{
Step I—Generate Population
'n' number of particles {p_1…..p_n} generated.

Step-II—Initialization
(Here, a particle represents a test case)
For each 'n' particle
{
Initialize position.
Initialize velocity
Set pos_velo_list(n); //for each particle.
For each 'n' particle
{
Identify local best.
}//end of for
Identify global_best particle (within search space).
//Selection of global best from 'n particles'
}//end of for

Step I: Here, 'n' numbers of particles/Test Cases generated. The generation of Test Cases can be done using Test Generation Tools.

Step-II: In this step, initialization of Test Cases done. Position & Velocity of particle helps in observing fitness of that particle. The best particles are selected from population.

Step-III: Each particle updates itself with best particle in population. The Position & Velocity get updated and fitness of each particle is either improved or retained.

Step-IV: Updated population is crossed over by dividing in two parts.

Step-V: Evaluate new off-springs and select the best off-spring as prioritized solution.

The whole process of hybrid prioritized algorithm carried out until feasible solution is achieved. The population keeps on updating to reach the destination (final solution).

Step-III--Updation/Comparison
For each 'n' particle
{
pos_velo_list (n); //read
sort_list (n); // (each 'n' particle compare its position w.r.t. local_best & then with global_best).
}//end of for

For each 'n' particle
{
sort_list(n); //read
For each 'n'
{

```
                updation_postion_velocity(n);
                //combine with either local or global best
                }//end of for

If(updated position& velocity > old position & velocity)
{
Keep the new position & velocity
}//end of if

Else {revert back to old position and velocity}
Generate 'New_Population';

        }//end of for
```

Step-IV—Crossover Operator

```
        Divide 'New_Population' into two set.
        Apply crossover operator on both parents.
        //two new off-springs generated.
        For each 'n'
        {
                Remove duplicity. /// comparison /////with in both
        n/2 +n/2 off-springs
        } // end for
```

Step-V—Examine Test-Suite

```
                Evaluate new off-springs//
//////if loop runs for both off-springs of length n/2 each

        If (any off-spring meets stopping criteria)
        {
                Set as final solution.
        }//end of if

        Else
        {
        Compare both off-springs
        Select best // near to stopping criteria
        Apply PSO on that off-spring
        (Go to Step-III).
        }//end of else

} //end of Proposed Algorithm
```

3.3 Explanation of Hybrid Prioritized Algorithm

Initially, particles are generated randomly. The particles are initialized with position and velocity. Each particle contains information regarding its own position and velocity. The position in our problem is either number of faults covered or number of conditional node covered and the velocity is either execution time or independent paths covered. Now, set fitness for each particle. The particle covering maximum number of faults/conditions in less time or maximum independent path covered is set as best particle. Each particle does comparison: its own position and velocity with best. On getting new position and velocity each particle compares it with old position and velocity respectively. The crossover operator is applied on new generated population such that: the off springs generated full fills the stopping criteria. The proposed Hybrid Prioritized algorithm has been automated for analyzing the test case prioritization. The algorithm has been implemented using JAVA in appropriate IDE.

3.4 Analysis of Hybrid Prioritized Algorithm

The analysis is based on complexity and correctness of algorithm [38]. The complexity depends on execution time and cost taken by algorithm whereas correctness of algorithm depends on modification-traversing and modification-revealing property of algorithm. The algorithm proved to select modification-revealing (fault-revealing) test cases of any program P, is said to be safe which means if any modification done to P, the faults will be detected by algorithm. The algorithm proposed in this research paper has been tested for code coverage testing where CFG (control flow graph) are used to test the program. The complexity of algorithm depends on number of calls made in algorithm and on data available. In proposed algorithm, the population is generated of size 'n' that takes O (n) operations. The initialization step three lists made that required n*[O (n) +n (2O (n))] operations. Thereafter, in updation and comparison of each particle w.r.t local best and global best comparison between particles takes O(n*logn) operation and updation for each particle is done. The crossover operator does n*O (n) operations for 'n' population. The last step executed for two off-springs of length n/2 each. In best case, the proposed algorithm's run time is O (n^3). But in worst case, the recursion operation is applied (in last step) and run maximum 2^n times. This makes complexity to reach O (2^n. n^3) which can make it NP complete. This algorithm has been executed on two approaches: fault-revealing and code coverage and it is analyzed that this algorithm works on best case prominently.

3.5 Fault based Testing Experimentation

To test perfectly, a program is tested for every single input, whether inputs are valid or invalid. The fault based testing can be used as quantifying measure for test cases. Test cases, which are not able to find mutant or "kill mutant" can be removed from test suite, that is, quality of test suite is tested and raised using mutation testing. Nevertheless, to execute mutation testing is quite expensive task, particularly on the larger applications.

For analyzing the hybrid prioritized algorithm, all examples are executed with implemented code (Triangle, Hotel, Student, Railway, and Quadratic). The examples are executed by crossing over each other and analyzing faults one-by-one. The algorithm is executed 25 times on the each problem and the best prioritized test suite is noted with its execution time. In hybrid prioritized algorithm, the result of each 25 runs is same; therefore, final optimized suite is represented in analysis table. It is declared above that best test suite should cover all faults with minimum time. The table shown below briefly explains the result of each run. The best test suite of each run is shown in Table-1.

Table 1: Summarized table of all examples for Crossover

Program Name	Priority Suite	Execution Time	Initial Test Suite Size	Average Build Time
Student	T8T6	18.33	9	3SEC
Hotel	T5T3	20.64	5	3SEC
Triangle	T5T2	7.0	17	4 SEC
Quadratic	T11T7T3	10.0	19	4SEC
Railway	T2T3	30.0	15	5SEC

With the table shown above it can be generalized that the priority test suite is the shorted time taken test suite. The student example takes T8T6 in '18.33' with average build of 3sec; hotel example takes T5T6 in '20.64' with average build of 3sec, triangle example takesT5T2 in'7.0' with average build of 4sec quadratic example takes T11T7T3 in '10.0' with average build of 4sec and railway example takes T2T3 in '30.0' with average build of 5sec. The effectiveness of run will be 100% as the best optimal test suite is retrieved with one run only. The saving % of crossover hybrid particle swarm optimization is same as of mutated hybrid particle swarm optimization as the number of test cases prioritized are same as of mutated one. Hence, the crossover hybrid particle swarm optimization algorithm is more efficient in getting optimal result.

3.6 Code Coverage Testing (Path Based) Experimentation

The code coverage based testing consists of: branch coverage, flow coverage, condition coverage. The path coverage testing is also a form of code coverage where internal structure of program is analyzed. In path coverage testing, the numbers of independent paths are searched (McCabe, 1976) based on control flow graph (CFG) that helps in finding the paths and flow of code. Each independent path assures covering at the least one condition.

According to McCabe, the basic path testing technique consists of for steps:

a) Computation of program graph.
b) Cyclomatic Complexity calculation.
c) Selection of Independent paths.
d) Generation of test cases covering each independent path.

The code coverage testing can be performed using: statement based coverage where statement's coverage within a code is considered, branch based coverage where branch's or decision's coverage within a code is considered, and condition based coverage where conditions within code are considered. All these coverage variations are metrics that are used to evaluate code coverage more efficiently. In the following example, a code-coverage testing criterion has been chosen to prioritize test cases and the conditional node coverage has been used as a parameter to achieve maximum path coverage where the independent paths are identified within program.The example takes three inputs and based on input values corresponding output is generated [39]. Test Cases chosen should be those that meet conditional node coverage criteria within the program to prioritize test cases. The test suite to be chosen as solution should consist of test cases covering all independent paths.

On the basis of program flow graph Cyclomatic complexity calculated is 7.

$V(G) = e - n + 2p$, where, $e = 23, n = 18, p = 1$

Test Cases are as follows:

T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19.

On the basis of DD-graph shown above, the Independent Paths are as follows:ABFGNPQR, ABCDEGHJKMQR, ABCDEGHIMQR, ABCDEGNOQR, ABCEGNPQR, ABCDEGHJLMQR, and ABFGNOQR

3.6.1 APPLYING HYBRID TECHNIQUE ON ABOVE EXAMPLE

According to PSO technique, each particle will look for global best and local best and will update its position and velocity for maximum conditional node and independent path coverage. The crossover operator is applied for swapping test cases to diversify search process.

STEP-I & II:

First, each particle will maintain its own position and velocity. Here, T2, T3, T5 T9, T10, T11, T15, T16 are global best particles as they cover maximum number of conditions. Then, each particle compares itself with best particle. Here, test case T1 has compared itself with T2 (local best) and global best {T2, T3, T5, T9, T10, T11, T15, T16} and since its local best is also one of test case in global best. It had updated its position and velocity w.r.t T2.

STEP-III:

Now, each particle will examine the updated position and velocity with respect to old position and velocity. The new values are retained if it improves position and velocity of particle otherwise revert back to old position.

STEP-IV:

Now, apply crossover operation on updated generation of population. In crossover operation, the population is divided in set of two parents. Then, two-point crossover is applied to search problem space more thoroughly. The one-point crossover is not suited for this problem as it was not diversifying the search space.

T10	T11	T12 T11	T13	T14 T15	T15	T16	T17	T18 T16	T19 T16

T1T2	T2	T3	T4	T5	T6T5	T7T5	T8T9	T9

Figure-1: Representing Crossover function

T1T2	T2	T12 T11	T13	T14 T15	T15	T16	T17	T18 T16	T19 T16

T10	T11	T3	T4	T5	T6T5	T7T5	T8T9	T9

Figure-2: two new off-springs generated.

STEP-V:

Now, the generated off-springs are compared to get best among them. They are compared on the basis of maximum conditional node coverage and independent path covered. Hence, the second table covers all independent paths and conditions. On removing duplicity: TEST-SUITE A: T10, T11, T3, T4, T5, T6, T7, T8, And T9

TEST-SUITE B: T1, T2, T12, T13, T16, T17, T18

TEST-SUITE B covers all independent paths. In this example, the total code coverage has been achieved using path coverage (for identifying independent paths of code).

4. COMPARISION

4.1 Fault Coverage Analysis

The fault detection examples are being compared on the parameter of: random order, reverse order, optimized order and prioritized order for better clarity in results. The comparison has been shown with the help of Average Percentage of Fault Detection (APFD) formula as defined by Rothermal [8]. This formula gives better vision towards final result. In below Sections the APFD of various orders are been calculated and represented with the help of bar graphs respectively.

4.1.1 Example-I

Table-2: Represents % of various orders of Triangle

Technique	APFD %
No Order	83.4
Random Order	84.3
Reverse Order	78.4
Optimal Order	90.3
Proposed Order	90.3

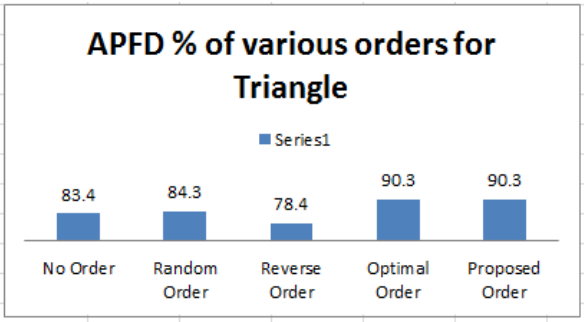

Fig-3: Represents APFD for prioritized triangle suite with other orders

4.1.2 Example-II

Table-3: Represents % of various orders of Hotel

Ordering of Test Cases	APFD%
No Order	46%
Random Order	66%
Reverse Order	62%
Optimum Order	66%
Prioritized Order	66%

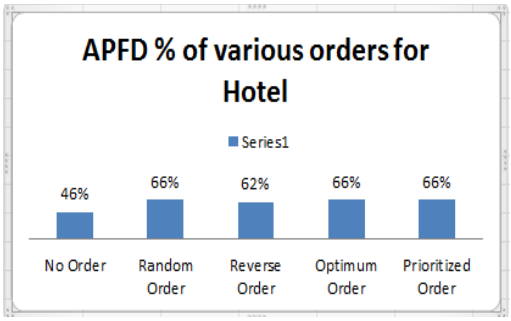

Fig-4: Represents APFD for prioritized hotel suite with other Orders

4.1.3 Example-III

Table-4: Represents % of various orders of student

Ordering of Test Cases	APFD %
No Order	76.8%
Random Order	76.8%
Reverse Order	70%
Optimum Order	78.9%
Prioritized Order	78.9%

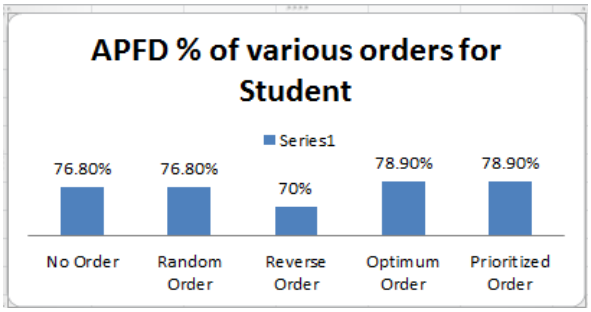

Fig-5: Represents APFD for prioritized student suite with other orders

4.1.4 EXAMPLE-IV

Table-5: Represents % of various orders of Railway

Ordering of Test Cases	APFD %
No Order	81.1
Random Order	71.6
Reverse Order	44.2
Optimal Order	87.9
Proposed Order	87.9

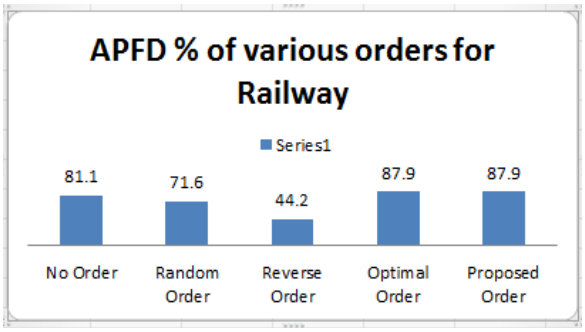

Fig-6: Represents APFD for railway prioritized suite with other orders

4.1.5 EXAMPLE-V

Table-6: Represents % of various orders for Quadratic

Ordering of Test Cases	APFD %
No Order	62.9
Random Order	82.8
Reverse Order	83.4
Optimal Order	90.3
Proposed Order	90.3

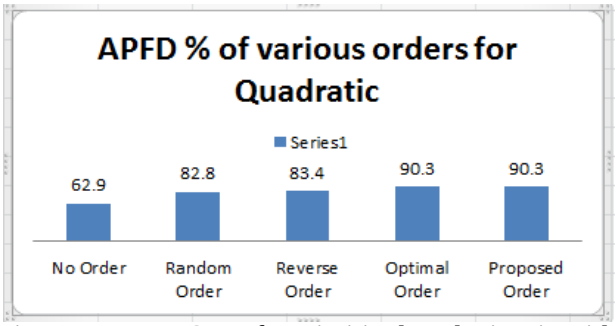

Fig-7: Represents APFD for prioritized quadratic suite with other orders

4.1.6 FINAL ANALYSIS OF FAULT COVERAGE:

The saving % achieved by proposed algorithm has been calculated:

$$S = \frac{\text{total test cases} - \text{prioritized test cases}}{\text{total test cases}} * 100 \qquad (2)$$

& the efficiency % has been calculated:

$$E = \frac{frequency\ of\ optimal\ test\ suite}{Total\ number\ of\ runs} * 100 \qquad (1)$$

Table-7: Represents summarized form of fault coverage examples

Programs	Fault Seeded	Effectiveness %	Saving %	Optimal APFD %	Prioritized APFD %
Student	5	28%	77.77778	78.9	78.9
Hotel	5	60%	60	66	66
Tri	6	16%	88.23529	90.3	90.3
Quad	9	20%	80	90.3	90.3
Railway	9	36%	89.47368	87.9	87.9

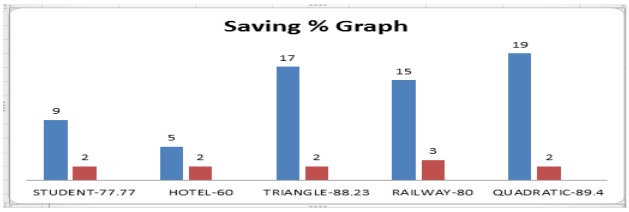

Fig-8: Representing the saving of test cases for execution

The Table-7 represents the summarized qualities of proposed algorithm's solutions for all examples. The table gives information regarding the number of faults seeded in each program respectively. The effectiveness of 25 runs for each program with the percentage of saving in test cases, then the optimal order and prioritized order (proposed order) are compared with each other for all examples respectively.

4.2 CODE COVERAGE ANALYSIS:

To analyze code coverage effectively the Average Percentage of Condition Coverage (APCC) approach has been used [99].

4.2.1 EXAMPLE-I

Table-8: Represents % of various orders of Triangle

Technique	APCC %
No Order	89.6
Random Order	88.7
Reverse Order	87.8
Optimal Order	93.1
Proposed Order	87.8

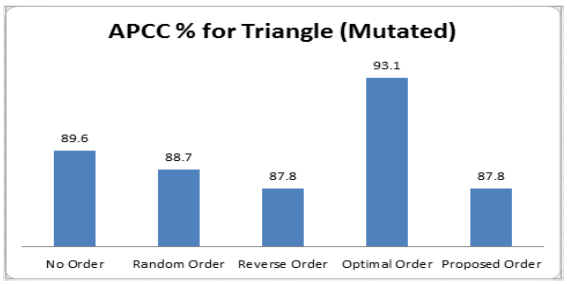

Fig-9: Represents APFD % for all orders

5. CONCLUSION:

The algorithm has been proposed to prioritize test cases using PSO technique along with crossover operator used in Genetic Algorithm. The technique is used to prioritize test cases on the basis of two different selection criteria namely: (i) total fault coverage with in time constrained environment, (ii) amount of code coverage. The results show that the proposed algorithm can do prioritization of test cases on different selection criteria other than used in this paper, as the algorithm uses the phenomena of convergence (PSO) while diversifying search space (GA operator) for regression testing. In proposed algorithm, PSO technique initializes particles to get fitness which further used for comparison within population to get local candidate solution. Thereafter, crossover operator has been used for diversifying the localization of solution by crossing over two parent strings with each other for getting two new off-springs which are compared according to desired stopping criteria. The effectiveness of proposed algorithm has been shown with the help of APFD and APCC values respectively. The APFD has been calculated for all examples for fault based testing and APCC has been calculated for code coverage testing example. The APFD and APCC calculation helps in evaluate usefulness of proposed algorithm. The APFD & APCC values are comparable w.r.t. optimal result, that proves algorithm prioritizes efficiently.

In this paper, test cases have been selected from large set of generated test data. The automation of prioritization has simplified the process. Further improvement is going on.

6. REFRENCES

[1] Alspaughy, S., Walcotty,K. R., Belanichz, M., Kapfhammerz,G. M., Lou Soffa,M.,2007 Efficient Time-Aware Prioritization with Knapsack Solvers, Proceedings of the ASE 2007 Atlanta, Georgia, November 2007.

[2] Garey,M.R., Johnson,D.S., 1979, Computers and Intractability, A Guide to the Theory of NP-Completeness, W. H. Freeman and Company, New York.

[3] Liang,Y, Liu,L.,Wang,D.,Wu, R., 2010,Optimizing Particle Swarm Optimization to Solve Knapsack Problem, ICICA, CICIS, Vol.105, Springer, Berlin, pp.: 437-443.

[4] Ezziane,Z., 2002 Solving the 0/1 knapsack problem using an adaptive genetic algorithm, Analysis and Manufacturing (AIEDAM), Vol.16(1), Jan-2010 pp.: 23-30.

[5] Premalatha,K., Natarajan, A.M.,2009Hybrid PSO and GA for Global Maximization,International Journal of Open Problems in Computer Science and Mathematics, Vol. 2, No. 4., pp. 597-608.

[6] Eberhart, R.C., Kennedy,J.,1995 A New Optimizer Using Particles Swarm Theory,IEEE Service Center, Piscataway, NJ, Nagoya, Japan, pp.:39-43.

[7] Srivastava, P. R., Kim,T., 2009 Application of Genetic Algorithm in Software Testing, International Journal of Software Engineering and Its Applications Vol. 3(4), pp.: 87-96.

[8] Lope, H.S.,Coelho, L.S., 2005 Particle Swarm Optimization with fast local search for the blind traveling salesman problem, Proceedings of Fifth International Conference on hybrid intelligent systems (HIS'05), Brazil, pp.: 245-250.

[9] Yoshikawa, M.,Nishimura, H., Terai, H., 2010 A New Genetic Coding for Job Shop Scheduling Problem Considering Geno type and Pheno type, Proceeding of the 4th WSEAS International Conference on Computer Engineering and Applications, Harvard University, Cambridge, SEAS Press, pp.: 59-62.

[10] Walcott, K. R.,Kapfhammer, G. M., Soffa, M. L., Roos, R. S., 2006 Time-aware test suite prioritization, Proceedings of International Symposium on Software Testing and Analysis, USA, pp. 1-19, July 2006.

[11] ASKARUNISA, A., SHANMUGAPRIYA, L., RAMARAJ, N., 2009 Cost and Coverage Metrics for Measuring the Effectiveness of Test Case Prioritization Techniques, pp.: 1-10.

[12] Wong, W. E., Horgan, J. R., London, S. and Agrawal, H., 1997, A study of effective regression testing in practice, In Proceedings of the 8th IEEE International Symposium on Software Reliability Engineering (ISSRE' 97), November 1997 , pp.: 264-274.

[13] Singh, Y., Kaur,A., Suri, B., 2006 A New Technique for Version – Specific Test Case Selection and Prioritization for Regression Testing, Journal of the Computer Society of India, Vol.36(4), pp.: 23-32.

[14] Aggrawal, K. K., Singh, Y., Kaur, A., 2004 Code coverage based technique for prioritizing test cases for regression testing, ACM SIGSOFT Software Engineering Notes , Vol.-**29(5)** pp.:1-4.

[15] Singh,Y., Kaur,A., and Suri,B., 2010 AHybrid Approach for Regression Testing in Intraprocedural Programs, JIPS, Vol.6 (1), pp.:21-32, March 2010.

[16] Singh,Y., Kaur,A., and Suri,B., 2010, "Test Case Prioritization Using Ant Colony optimization", Association in Computing Machinery,ACM SIGSOFT Software Engineering Notes, USA, July 2010, pp.: 1-7.

[17] Hla, K. H.S., Choi, Y., Park, J. S., 2008 Applying Particle Swarm Optimization to Prioritizing Test Cases for Embedded Real Time Software Retesting, Proceedings of the IEEE 8th International Conference on Computer and Information Technology Workshops, pp.: 527-532.

[18] Aggrawal, K. K., Singh, Y., Kaur, A., 2004 Code Coverage Based Technique for prioritizing Test Cases for Regression Testing, ACM SIGSOFT Software Engineering Notes, Vol.29(5), September 2004.

[19] Fischer, K., Raji, F.,Chruscicki, A., 1981 A methodology for retesting modified software, In *Proc. of the Nat'l. Tele. Conf. B-6-3*, Nov. 1981, pp. 1-6.

[20] Mehta, A., Heineman,G.T., 2000 Evolving Legacy Systems Features for Regression Test Cases and Components, Worcester Ploy technique Institute, MA, pp.: 1-11.

[21] S. Yoo, M. Harman and S. Ur, " Highly Scalable Multi Objective Test Suite Minimization Using Graphic Card", Department of Computer Science, University College of London, UK, 2007, pp.: 1-26.

[22] Harman, M., Mansouri, A., 2010 Search Based Software Engineering: Introduction to special issue of IEEE Transactions on Software Engineering, IEEE Transactions, Vol-6, No-4, Nov-2010, pp.: 737-741.

[23] M. Harman, "Making the case for MORTO: Multi objective Regression Test Optimization", University College of London, CREST center London, pp.:1-4

[24]Li,Z., Harman, M. and Hierons, R. M., 2007 Search algorithms for regression test case prioritization, *IEEE Trans. On Software Engineering*, Vol.-**33**, No.-**4**, April 2007, pp.: 76-89.

[25] Liu, H., Sun, S., Abraham, A.,2006 Particle swarm approach to scheduling work-flow applications in distributed data- intensive computing environments, Proceedings of Sixth International Conference on In Intelligent Systems Design and Applications, ISDA'06, pp.: 661–666.

[26] Zhao, F., Zhang, Q., Yang, Y., 2006 An improved particle swarm optimization based approach for production scheduling problems, in: Proceedings of the IEEE International Conference on Mechatronics and Automation, pp.:2279–2283.

[27] Kong,X., Sun, J., Xu,W., 2006 Particle swarm algorithm for tasks scheduling in distributed heterogeneous system, in Proceedings of Sixth International Conference on ISDA'06, pp.: 690–695.

[28] Laskari,E.C., Meletiou, G.C., Vrahatis, M.N., 2006 Utilizing evolutionary computation methods for the design of s-boxes, in Proceedings of International Conference on CIS-2006, pp.: 1299–1302.

[29] Zhi,X.H., Xing, X.L., Wang, Q.X., Zhang, L.H., Yang, X.W., Zhou, C.G., Liang, Y.C., 2004 A discrete PSO method for generalized TSP problem, in: Proceedings of International Conference In Machine Learning and Cybernetics, (4), pp.: 2378–2383.

[30] Liu,D.S., Tan,K.C., Goh,C.K., Ho, W.K., 2006 On solving multi objective bin packing problems using particle swarm optimization, in: Proceedings of IEEE Congress on Evolutionary Computation, CEC2006, pp.: 2095–2102.

[31] Hou,Y.,Zhao, C., Liao,Y., 2006 A new method of test generation for sequential circuits, in Proceedings, 2006 International Conference on Communications, Circuits and Systems, pp.: 2181–2185.

[32] Sheta, A., 2006 Reliability growth modeling for software fault detection using particle swarm optimization, Proceedings of IEEE Congress on Evolutionary Computation, CEC2006, pp.: 3071–3078.

[33] Krishnamoorthi, R., Mary, S.A., 2009Regression Test Suite Prioritization using Genetic Algorithms, International Journal of Hybrid Information Technology, Vol.2(3),, July 2009, pp.: 35-52.

[34] Kamble, A., 2010 Incremental Clustering in Data Mining using Genetic Algorithm, International Journal of Computer Theory and Engineering, Vol.2(3), , June 2010, pp.: 1793-8201.

[35] Bergmann,K. P., Scheidler,R., Jacob, C., 2008Cryptanalysis using genetic algorithms, in 10th annual conference on Genetic and evolutionary computation, ACM New York, NY, USA, pp.: 1099-1100.

[36] Krishna,K. S. R.,Reddy, A. G., Prasad, M.N.G., Rao, K.C., Madhavi, M., 2010 Genetic Algorithm Processor for Image Noise Filtering Using Evolvable Hardware, International Journal of Image Processing, Vol.4(3), pp.: 240-250.

[37] Li,M., Zhang,Y., Jiang, W., Xie,J., Coll,Z., 2009 A Particle Swarm Optimization Algorithm with Crossover for Resource Constrained Project Scheduling Problem, Proceedings of IITA SSME'09, pp.: 69-72.

[38] Rothermel,G.,Harold, M. J., 1997 A Safe, Efficient Regression Test Selection Technique,ACM Transaction on Software Engineering and Methodology, Vol.6 (2),, April 1997, pp.: 173-210.

[39] Aggarwal,K.K.,Singh,Y., 2001 Software Engineering, New Age International (P) Ltd.; Publishers, 4835/24, Ansari Road, Daryaganj, New Delhi.