# Hybrid Particle Swarm Optimization for Regression Testing

Dr. Arvinder Kaur

University School of Information Technology,
GGSIPU,
Delhi, India

Divya Bhatt

University School of Information Technology,
GGSIPU,
Delhi, India

**Abstract— Regression Testing ensures that any enhancement made to software will not affect specified functionality of software. The execution of all test cases can be long and complex to run; this makes it a costlier process. The prioritization of test cases can help in reduction in cost of regression testing, as it is inefficient to re- run each and every test case. In this research paper, the criterion considered is of maximum fault coverage in minimum execution time. In this research paper, the Hybrid Particle Swarm Optimization (HPSO) algorithm has been used, to make regression testing efficient. The HPSO is a combination of Particle Swarm Optimization (PSO) technique and Genetic Algorithms (GA), to widen the search space for the solution. The Genetic Algorithm (GA) operators provides optimized way to perform prioritization in regression testing and on blending it with Particle Swarm Optimization (PSO) technique makes it effective and provides fast solution. The Genetic Algorithm (GA) operator that has been used is Mutation operator which allows the search engine to evaluate all aspects of the search space. Here, AVERAGE PERCENTAGE OF FAULTS DETECTED (APFD) metric has been used to represent the solution derived from HPSO for better transparency in proposed algorithm.**

***Keywords: Regression Testing, Particle Swarm Optimization, Genetic Algorithm.***

## I. INTRODUCTION

The assurance regarding any modification in the software is given by the process called regression testing. It makes sure that alterations have not affected functional characteristics of software. It is quite expensive technique to be used. Some techniques such as test selection [1], test prioritization [1] has been proposed by researchers for effective cost reduction in regression testing. For prioritization in regression testing, the ordering of test cases is done priority-wise such as: highest priority test case is to be executed first and so on, according to selected criteria, e.g. to increase fault rate detection, to maximize code coverage. The test case prioritization in time-constrained environment can be reduced to NP complete 0/1 knapsack problem [2], [3]. The NP complete knapsack problem has been solved by PSO very effectively [4]. It has been proved that test case prioritization in software testing is NP complete problem [1]. To reduce it into polynomial time, time based constraint has been introduced in this research work. The fault-coverage efficiency of test cases is more critical to exploit, as one test case cannot cover all faults. Here, test-suite is built consisting of test cases that are full-filling the criteria.

The Hybrid Particle Swarm Optimization (HPSO) can be used to solve problems of time-constrained environment efficiently. The PSO has been used earlier to solve traveling salesman problem [4], decision making [6], and speech coding [5]. The PSO is generally the concept taken from birds, flying in search of destination containing good quality food. They form a typical shape, (mostly triangular) while flying in group. This is because, each individual look for its best position so that they don't lack behind in the group. In this process, they converge towards that best position periodically. Whereas, in GA, the chromosomes breed with each other to reproduce new off-springs that are more close to the solution and are fitted to reproduce further. The whole concept of PSO and GA are to be used in the regression testing to make it quick, cost efficient and more scheduled. The Average Percentage of Faults Detected (APFD) has been used widely for evaluating test prioritization techniques [1].

## II. RELATED WORK

Regression Testing has been solved and explored in many ways. Many techniques are proposed by researchers for reducing cost related to regression testing. That includes [7], [8] regression test selection, regression test prioritization, and hybrid approach. In selective regression testing technique, only a subset of test cases is selected and re run. The regression testing has been solved using GA [25], ACO [9] and from many more. In the following techniques the regression testing has been analyzed and optimized in polynomial time constraint. On being an optimization approach PSO has been used to solve routing optimization [10] where the optimization of route is done by analyzing the shortest route in the search area, job-scheduling [11] optimization deals with appropriate scheduling of jobs of given problem in optimal time limit, network security [12] which consist of policies and activities for more secured network, cryptography and crypt-analysis [13] that enables data/message security using different algorithms, software faults detection [14] deals with prediction of fault in the software. Dynamic clustering [15], fuzzy clustering [16] these clustering methods are applied on data for intelligent separation of informational data from others, (TSP) traveling salesman problems [17] is NP problem that is needed to be solved in polynomial time limit, similar to TSP, the packing and knapsack [18] is NP problem solved using PSO and other i.e. minimum spanning trees [19]. The path optimization [20], vehicle routing [21], [22] are combinatorial optimization problems same like TSP where optimal path from given search space is formulated. The GA is favorable field of many scientists, explorers and researchers; this makes application of GA in the diverse fields such as: Robotics [23] for path planning or decision making, Optimized Telecommunicating Routing [24] where shortest route which is cost effective is analyzed, Regression Testing [25] to test whether modification has adverse effects on software, and Data Mining [26] for useful information digging. The GA is efficiently been used in the field of encryption and code breaking [27] for ethically decoding the information for military or government usage.

## III. PARTICLE SWARM OPTIMIZATION

In PSO (Particle Swarm Optimization), a search space has been explored for problems and parameters are identified which maximizes aim of that given problem. In PSO, "n" particles and position of each particle stands a potential solution of given problem. In 1995 [28], James Kennedy and Russell C. Eberhart gave this technique, by drawing two individual concepts of PSO:

1. By observing swarming habits of special kinds of animals (flock of birds, school of fishes), derived the thought of swarm intelligence and,
2. Evolutionary Computing Algorithms.

The whole procedure begins by initializing population (a class of particles) that promises a potential solution and thence, optimal solution searched by position and velocity updation of particles [29]. Each particle in a given population has been assigned position and velocity of its own. The updation and modification is carried out to acquire "best" or optimal solution for the stated problem. With each iteration, values that are "best" for particle (individually) are identified based on: the value that is best achieved so far by a particle (pbest), whereas the best value incurred by some other particle with in population (gbest) and the best value achieved by immediate neighbors (local best) of a particle (lbest).To support the above principle following equations represents modification in velocity of particle which further causes position updation [30]:

$$V_{ik+1} = w*V_{ik} + c_1\, rand_1 (\ldots) *(pbest_i - s_{ik}) + c_2\, rand_2 (\ldots) * (gbest - s_{ik}) \quad (1)$$

Where, $v_{ik}$ : velocity of agent i at iteration
w: weighing function,
$c_j$ : weighting factor (uniformly distributed random number between 0 and 1),
$s_{ik}$ : current position of agent i at iteration k,
$pbest_i$ : pbest of agent i,
gbest: gbest of the group.
And, the weighting function is calculated as:

$$w = wMax - [(wMax - wMin) \times iter]/maxIter \quad (2)$$

where, wMax= initial weight,
wMin = final weight,

maxIter = maximum iteration number,
iter = current iteration number.

$$s_{ik}+1 = s_{ik} + V_{ik}+1 \quad (3)$$

In whole, the PSO is an optimization approach that updates and validates each step. The process of updation comes from solution space only. A particle updates itself depending on the global best or local best. The random variable chosen is not generated but formulated by analyzing the population's position & velocity and particle's position & velocity.

## IV. GENETIC ALGORITHM

The theory of "natural evolution" in the origin of species was stated by Charles Darwin. Over several years, organisms evolved believing on "survival of fittest" principle [31].

In 1975, Holland developed idea of Genetic Algorithm in the book "Adaptation in Natural and Artificial System", he described how to apply natural evolution principle for optimization of problems and first GA was built, based on genetics and evolution [32]. Now, GA is used to resolve complex optimization problem, like job-scheduling, games playing, time tabling [32], [34]. The word "genetics" is derived from Greek word "genesis" meaning "to grow" or "to become" [32].

GA converts possible solution of a problem in a genome or chromosome, string like structure, then applies GA operators such as [30], [34]: selection, recombination (crossover) and mutation to these genomes. Hence, preserves decisive information. The execution of GA starts with generation of random population of genomes or chromosomes. Then starts the evaluation of population by assigning reproductive chances in such a manner that genome or chromosome or individual representing optimal solution for a problem is given more opportunity to grow (reproduce) than other [30].

The process of GA can be described as [30]:

- Initialization of population: The population is randomly generated to produce the solution of a given problem.
- Selection: This is the phase where an individual is chosen for reproducing better solution. The initialization of fitness function [30] is major part, helping in evaluating each individual and decides whether it is fit for generating intermediate population or not. Hence, selection procedure depends hugely over fitness function for better population.
- Crossover: It is also called as recombination. Here, two parent strings are swapped with each other to reproduce new promising off-springs.
- Mutation: It consists of changing values of chromosomes or genomes by mutating them with each other. The mutation rate to be applied can be less than 1% probability.

Thus, the process of evolution of population, selection, crossover and mutation generates new off-springs with best solution.

## V. HYBRID PARTICLE SWARM OPTIMIZATION

While searching for best solution in the search space, the particle can converge to one point between local and global best, hence, resulting in single point convergence, ignoring other aspects of search spaces. While searching for best solution in search space, particles communicate with each other and flow information about position and velocity. Moreover, the flow of information within similar particles can result in loss of diversity. Hence, this can increases possibility of getting trapped in local optima.

To overcome this limitation, GA factor has been introduced in PSO concept. The reason to apply mutation [35] in PSO is to increase population diversity and ability of PSO to avoid converging into one point of solution [33].

### A. *Assumptions:*

1. Initial population is randomly generated.
2. A particle in population depends on problem (test cases).
3. The position and velocity depends on problem.
4. Stopping criteria is needed to be fulfilled.

**B.** *Proposed Algorithm:*

Step I: Generation: 'n' particle population generated.
Step II: Initialize:
For each particle
{
Initialize position of each particle.
Initialize velocity of each particle.
} // end of for
Step III: Mutation:
For each particle
{
Randomly chose one particle among population.
Mutate each population with chosen mutant.

} // end of for
Step IV: Updation:
For each particle
{
If (updated position & velocity > old)
Update position of each particle.
Update velocity of each particle.
Else
Revert to old position.} // end of for
Step V: Examine:
Check each particle.
If (any particle fulfills stopping criteria)
STOP.
Else
Go to Step III.
Step VI: End of algorithm Proposed.

**C.** *Flowchart:*

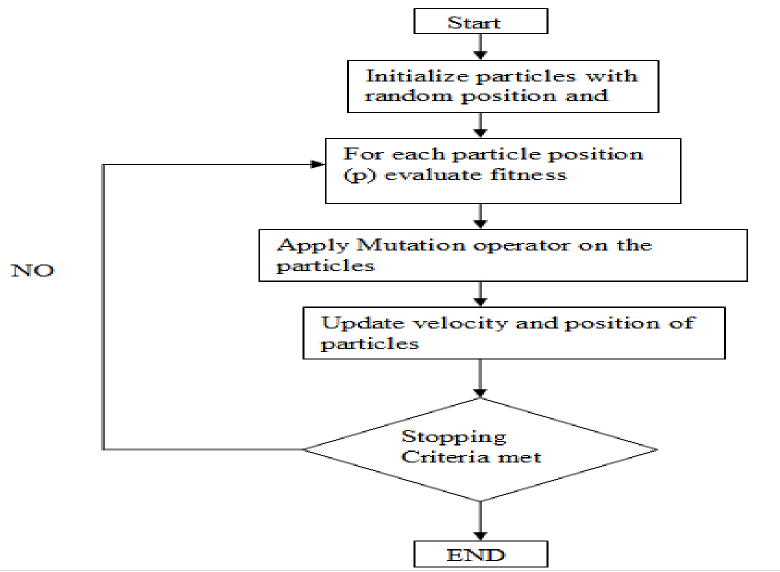


Figure 1 Figure represents flow of proposed HPSO algorithm

### D. Explanation of Algorithm:

The initiation of HPSO algorithm starts with generation of population. The HPSO is an artificial intelligence based technique, where we consider some population. The population plays major role in deciding which way solution will approach. The population consists of particles or group of particles forming population. On selecting the population randomly from a given problem, we assign them position and velocities on the basis of original PSO technique. In our process of test case prioritization, we will consider position of particle as number of faults covered by that particle and velocity of the particle as total execution time taken by it in covering the faults. The regression testing which will be followed in example is total fault covered – in less time. The stopping criterion is to be decide, on the basis of which HPSO algorithm will end. The stopping criteria can be: maximum number of iterations, full optimized solution or population is not fit to generate good solution. The criteria considered here is: full optimized solution. While criteria dose not meet, the mutation [34] and updation process will be carried on. In mutation, the random GA mutation theory has been used, where a random particle from existing particles is chosen and added in rest some of selected particles. On mutating the particles, the positions and velocities are updated. The best test suite with maximum fault and less time taken is taken as global best. While each individual particle's position and velocity is updated by comparing with its old position and velocity. The updated position and velocity will be retained if improvement is measured otherwise revert back to old position and velocity. This process will continue until the stopping criterion is met.

The proposed HPSO algorithm has been automated for analyzing the test case prioritization. The algorithm has been implemented using JAVA in appropriate IDE.

### E. Problem Formulation:

The test case prioritization technique's basic evaluation is to have maximum number of faults covered in minimum number of test cases required. Here, we are analyzing the execution time of every test case. The fault measuring technique used is fault coverage based testing technique. In this example, there are test cases forming Test Suite (TS) = {T1, T2, T3, T4, T5, T6, T7, T8, T9} and the faults covered by those test cases are represented as Faults Covered (FC) = {F1, F2, F3, F4, F5}. It has been represented in TABLE-1 below as:

TABLE 1 TEST CASES WITH NUMBER OF FAULTS AND EXECUTION TIME TAKEN

| Test Cases ID | Total Faults Covered | Execution Time taken |
|---|---|---|
| T1 | 4 | 11.5 |
| T2 | 2 | 11.5 |
| T3 | 3 | 12.33 |
| T4 | 3 | 10.66 |
| T5 | 1 | 15 |
| T6 | 3 | 8.33 |
| T7 | 1 | 15 |
| T8 | 3 | 10 |
| T9 | 1 | 11 |

On getting the number of faults identified by each test case, we need the execution time required for those test cases and the total number of faults covered in that time.

Now we will apply hybrid particle swarm optimization (HPSO) technique for solving the test case prioritization problem. According to it, we need a stopping criterion, in our example, our stopping criteria will be fully optimized solution with maximum number of fault covering in a test suite, and objective functions is to be fulfilled are: maximum fault covered in less execution time taken.

In particle swarm optimization, a particle updates its position and velocity on the basis of its neighbor's best and the whole flock updates its position based on total best.

Now applying Hybrid PSO for test case prioritization technique, the following iteration will occur:

For k=0

Here random number is T3. In TABLE-2 first iteration has been shown where updation depending on mutant generated is done.

TABLE 2 FIRST ITERATION CHANGE

| New Test case ID | Test cases ID | Faults covered | New execution time |
|---|---|---|---|
| E1 | T1 T3 | 4 | 23.83 |
| E2 | T2 T3 | 4 | 23.83 |
| E3 | T3T3 | 3 | 12.33 |
| E4 | T4 T3 | 4 | 22.99 |
| E5 | T5 T3 | 3 | 27.33 |
| E6 | T6T3 | 4 | 20.66 |
| E7 | T7T3 | 3 | 27.33 |
| E8 | T8T3 | 3 | 22.33 |
| E9 | T9T3 | 3 | 23.33 |

In TABLE-2, with every particle T3 is added, which will change each particles position and velocity. On the basis of this global best and individual best is chosen in updation process.

Updating: In this process the each particle compare its modified position with previous position if previous position was better than previous will be kept otherwise new updated position will be kept. TABLE-3 shows the updated process. The improved positions retained only.

TABLE 3 AFTER UPDATING POSITION & VELOCITY

| NEW Test case ID | Test cases ID | Faults covered | New execution time |
|---|---|---|---|
| E1 | T1 | 4 | 11.5 |
| E2 | T2 T3 | 4 | 22.8 |
| E3 | T3 | 3 | 12.33 |
| E4 | T4 T3 | 4 | 22.99 |
| E5 | T5 T3 | 3 | 27.33 |
| E6 | T6T3 | 4 | 22.33 |
| E7 | T7 | 3 | 8.33 |
| E8 | T8T3 | 3 | 27.33 |
| E9 | T9T3 | 3 | 23.33 |

For k=1

Here random number is T4. In previous iteration all faults have not been covered. Therefore, random number is generated again. And updation in previous position is done depending on improvement.

TABLE 4 SECOND ITERATION CHANGE

| **New Test case ID** | **Test cases ID** | **Faults covered** | **New execution time** |
|---|---|---|---|
| E1 | T1T4 | 5 | 22.16 |
| E2 | T2 T3T4 | 5 | 32.99 |
| E3 | T3T4 | 4 | 22.99 |
| E4 | T4 T3T4 | 4 | 22.99 |
| E5 | T5 T3T4 | 4 | 37.99 |
| E6 | T6T3T4 | 4 | 32.99 |
| E7 | T7T4 | 4 | 18.99 |
| E8 | T8T3T4 | 4 | 37.99 |
| E9 | T9T3T4 | 4 | 22.99 |

In Table-4, with every particle T4 is added, which will change each particles position and velocity. On the basis of this global best and individual best is chosen in updation process.

Hence, on the basis of stopping criteria the new table with prioritization of test cases will be generated as shown in TABLE-5.

TABLE 5 FINAL TABLE FOR PROBLEM

| **New Test case ID** | **Test cases ID** | **Faults covered** | **New execution time** |
|---|---|---|---|
| E1 | T1T4 | 5 | 22.16 |
| E2 | T2 T3T4 | 5 | 32.99 |
| E7 | T7T4 | 4 | 18.99 |
| E3 | T3T4 | 4 | 22.99 |
| E4 | T4 T3T4 | 4 | 22.99 |
| E9 | T9T3T4 | 4 | 22.99 |
| E6 | T6T3T4 | 4 | 32.99 |
| E8 | T8T3T4 | 4 | 37.99 |
| E5 | T5 T3T4 | 4 | 37.99 |

Therefore, after updating the last iteration we get the best test suite of following priority:

The test suite {T1, T4} with minimum execution time **'22.16'**, in the unit of seconds has been selected.

In starting off, each iteration, we apply mutation operator. In which, a random number is taken and added in sequence of test cases. The Table-5 is the final prioritized table on the basis of hybrid PSO. The test suites arranged are on the fitness function basis defined previously in algorithm.

### F. Analysis of Algorithm:

The proposed algorithm HPSO has been executed on the above example and on two more to testify correctness of algorithm. The average running time of algorithm on IDE is less than 6 units of time in all. The optimal result comes in 5 re-run of algorithm in average. The number of iteration required is usually one. The algorithm proposed in this research paper executes in polynomial time with complexity of $O(n^2)$ where, n is the population generated. Hence, the result of HPSO after solving above problem can be represented through APFD representation. The APFD Percentage as calculated by concerning test suite selected from above program solution [36]. The APFD is given as:

$$1 - \frac{TF1 + TF2 + \ldots\ldots + TFm}{nm} + \frac{1}{2n} \quad (4)$$

Where, T - Test Suite to be analyzed
m -Number of faults
n - Number of test cases
$TF_i$ - Position of the first test with in test suite for $i^{th}$ fault.

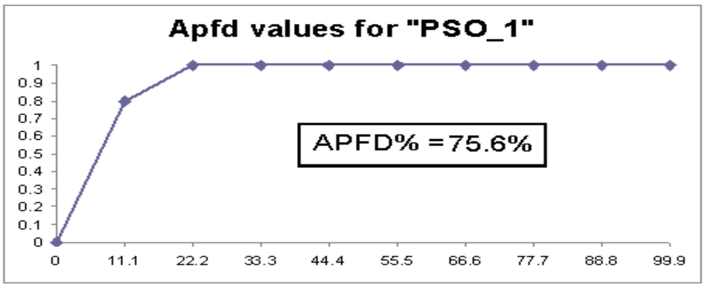


Figure 2 Figure representing APFD chart for H-PSO on above problem

This results that 75.6% of fault coverage has been done by HPSO.

### G. Threats to Validity:

The HPSO algorithm proposed here has been executed and following areas have been detected as threat to validity.

1. The optimal result depends on observing the final result.
2. The HPSO depends on randomly generating a mutant that makes the execution time quite long.
3. The algorithm has been tested on less number of programs. More analysis is needed.

## VI. CONCLUSION

This paper presents a combined analytic view of evolutionary computation techniques namely Genetic Algorithm and Particle Swarm Optimization. The PSO is an optimization technique, where the global solution is constructed by the analysis of the local optimal solution. The Genetic Algorithm have been discussed which produces the fittest population from the current population using genetic operators to generate new and improved population. These techniques have immense potential and scope of application ranging from engineering to software engineering, optimization problems to non-optimization problems, fuzzy to Neuro-fuzzy, robotics, electronics and many more. In this paper we presented, the hybrid approach to solve prioritization in regression testing. The execution of algorithm has shown the effectiveness of the technique proposed. The automation of algorithm has provided solid base for its effectiveness. The analysis of algorithm shows the percentage of faults detected in prioritized test suite with the help of APFD.

## REFERENCES

[1] Y. Singh, A. Kaur. and B. Suri: “A New Technique for Version – Specific Test Case Selection and Prioritization for Regression Testing”. Journal of Computer Society of India, 36(4), pp. 23-32., 2006
[2] M. R. Garey and D. S. Johnson. “Computers and Intractability: A Guide to the Theory of NP-Completeness”. W. H. Freeman & Co., New York, NY, USA, 1979.
[3] G. Rothermel, R. J. Untch, and C. Chu. “Prioritizing test cases for regression testing”. IEEE Trans. on Softw. Eng., vol-27(10): pages:929-948, 2001.
[4] Yi-Chun Xu and Ren-Bin Xiao. “Hybrid particle swarm algorithm for packing of unequal circles in a larger containing circle”. In Intelligent Control and Automation, 2006. WCICA 2006. The Sixth World Congress on, pages3381–3385,2006
[5] G. K. Venayagamoorthy and W. Zha., “Comparison of non uniform optimal quantizer designs for speech coding with adaptive critics and particle swarm”. 2007, IEEE Transactions on Industry Applications , vol-43, pages: 238–244, 2007

[6] J.Nenortaite and R.Simutis. “Adapting particle swarm optimization to stock markets. In Intelligent Systems Design and Applications”, 2005. ISDA’05. Proceedings. 5th International Conference on, pages:520–525,2005.
[7] Y. Singh, A. Kaur, B. Suri, “A New Technique for Version – Specific Test Case Selection and Prioritization for Regression Testing” Journal of the CSI, pp.: 23-32, 2006.
[8] Yogesh Singh, Arvinder Kaur and Bharti Suri, “A Hybrid Approach for Regression Testing in Intraprocedural Programs”, Journal of Information Processing Systems, Vol.6, No.1, pp.:21-32, 2010.
[9] Y. Singh, A. Kaur, B. Suri, “Test Case Prioritization Using Ant Colony optimization”, Association in Computing Machinery ,Newsletter ACM SIGSOFT Software Engineering Notes, New York, NY, USA, pp.: 1-7, July 2010.
[10] C. Ji,Y. Zhang,S. Gao,P. Yuan, Z. Li. “Particle swarm optimization for mobile ad hoc networks clustering”, in: Proceedings of the IEEE International Conference on Networking, Sensing and Control, Vol.1,, pp.: 372-375, 2004.
[11] S. Srinoy , W. Kurutach, “Combination artificial ant clustering and k-PSO clustering approach to network security model”, in: Proceedings of International Conference on Hybrid Information Technology, ICHIT’06. Vol-2., pp.: 128–134, 2006
[12] E.C. Laskari, G.C. Meletiou, M.N. Vrahatis, “Utilizing evolutionary computation methods for the design of s-boxes”, in Proceedings of International Conference on Computational Intelligence and Security, pp.: 1299–1302, 2006
[13] A.Sheta, “Reliability growth modeling for software fault detection using particle swarm optimization”, in Proceedings of IEEE Congress on Evolutionary Computation, CEC2006, pp.: 3071–3078, 2006.
[14] J. H. Lin, T. Y. Cheng, “Dynamic clustering using support vector learning with particle swarm optimization”, in Proceedings of 18th International Conference on Systems Engineering, 2005. ICSEng’2005, pp.: 218–223, 2005
[15] L. Wang, Y. Liu, X. Zhao, Y. Xu, “Particle swarm optimization for fuzzy c-means clustering”, in Proceeding of the Sixth World Congress on Intelligent Control and Automation, WCICA -2006, pp.: 6055–6058, 2006.
[16] X.H. Zhi, X.L. Xing, Q.X. Wang, L.H. Zhang, X.W. Yang, C.G. Zhou, Y.C. Liang, “A discrete PSO method for generalized TSP problem, in: Proceedings of International Conference on In Machine Learning and Cybernetics”,Vol-4, pp.: 2378–2383. 2004.
[17] Y.C. Xu, R. B. Xiao, “Hybrid particle swarm algorithm for packing of unequal circles in a larger containing circle”, in Proceedings of the Sixth World Congress on In Intelligent Control and Automation, WCICA 2006, pp.: 3381–3385, 2006.
[18] E.F.G. Goldbarg, G.R. de Souza, M.C. Goldbarg, “Particle swarm optimization for the bi-objective degree constrained minimum spanning tree”, in: Proceedings of IEEE Congress on Evolutionary Computation, 2006. CEC 2006, pp.: 420–427, 2006.
[19] G. Y. Zhu, “Drilling path optimization based on swarm intelligent algorithm”, in: Proceedings of IEEE International Conference on Robotics and Bio-mimetic ROBIO’06, pp.: 193–196, 2006
[20] B. Wu, W. Wang, Y. Zhao, X. Xu, F. Yang, “A novel real number encoding method of particle swarm optimization for vehicle routing problem”, in Proceedings of The Sixth World Congress on In Intelligent Control and Automation, WCICA 2006, pp.: 3271–3275, 2006.
[21] B. Wu , Y. Zhao, Y. Ma, H. Dong, W. Wang, “Particle swarm optimization method for vehicle routing problem”, in Proceedings of Fifth World Congress on Intelligent Control and Automation, WCICA 2004, Vol-3, pp.: 2219–2221, 2004.
[22] S. Cernic , E. Jezierski , P. Britos , B. Rossi, R. García Martínez , “Genetic Algorithms Applied to Robot Navigation Controller Optimization”, in: Proceedings of the International Conference on Intelligent Systems and Control, Páginas, Santa Bárbara, California, pp.: 230-234, 1999.
[23] N.K. Cauvery, K. V. Viswanatha, “Routing in Dynamic Network using Ants and Genetic Algorithm”, International Journal of Computer Science and Network Security, Vol-9 No-3, pp.: 194-200, ( March 2009).
[24] R. Krishnamoorthi , S.A. Sahaaya Arul Mary, “Regression Test Suite Prioritization using Genetic Algorithms”, International Journal of Hybrid Information Technology, 2(3), pp.: 35-52, July, 2009.
[25] A. Kamble , “Incremental Clustering in Data Mining using Genetic Algorithm”, International Journal of Computer Theory and Engineering, 2(3), pp.: 1793-8201, June 2010.
[26] K. P. Bergmann, R. Scheidler, C. Jacob, Cryptanalysis using genetic algorithms, in: Proceedings of the 10th annual conference on Genetic and evolutionary computation, GECCO '08, ACM New York, NY, USA, pp.: 1099-1100, 2008.
[27] J. Kennedy and R. C. Eberhart., Swarm Intelligence, Morgan Kaufmann Publishers, San Francisco, (2001).
[28] Frans van den Bergh. An Analysis of Particle Swarm Optimizers, PhD thesis, University of Pretoria, 2001.
[29] P.J. Angeline, "Using Selection to Improve Particle Swarm Optimization", IEEE int. Conf., , pp. 84-89, 1998.
[30] M. Mitchell. An Introduction to Genetic Algorithms . MIT Press, Cambridge, MA, 1998.
[31] S.N. Sivanandam, S.N. Deepa., Introduction to Genetic Algorithms, Springer, Berlin, 2008.
[32] M. Jancic, and I. Grundy, "Particle swarm optimization with mutation," in Proc. Congr. Evol. Compt., vol. 2, pages. 1425 – 1430, 2003.
[33] D E Goldberg, Genetic Algorithms in search, optimization, and machine learning, Addison-Wesley Publishing Corporation, Inc, 1989.
[34] S. Andrews, "An Investigation into Mutation Operators for Particle Swarm optimization," in Proc. Congr. Evol. Compt., pages: 1044 – 1051, 2006.
[35] K. Premalatha and A.M. Natarajan, “Hybrid PSO and GA for Global Maximization”, Kongu Engineering College, Erode, pages: 598-608, 2009.
[36] K. R. Walcott, G. M. Kapfhammer, M. L. Soffa, and R. S. Roos, “Time-aware test suite prioritization”, International Symposium on Software Testing and Analysis, Association in Computing Machinery, Portland, Maine USA, pp. 1-19, 2006.

**AUTHORS PROFILE**

**Dr. Arvinder Kaur**

Dr. ArvinderKaur is an Associate Professor with the University School of Information Technology, Guru Gobind Singh Indraprastha University, India. She obtained her doctorate from Guru Gobind Singh Indraprastha University and her master’s degree in computer science from Thapar Institute of Engineering and Technology. Prior to joining the school, she worked with B.R. Ambedkar Regional Engineering College, Jalandhar and Thapar Institute of Engineering and Technology. She is a recipient of the Career Award for Young Teachers from the All India Council of Technical Education, India. Her research interests include software engineering, object-oriented software engineering, software metrics, software quality, software project management, and software testing. She also is a lifetime member of ISTE and CSI. Kaur has published 60 research papers in national and international journals and conferences.

**DIVYA BHATT**

She has done MCA from Guru Gobind Singh Indraprastha University in 2009 and currently pursuing M.Tech (IT) from Guru Gobind Singh Indraprastha University. Her area of interest is Software Engineering.