*International Journal of Engineering & Technology*, *7 (1.1) (2018) 431-434*

**International Journal of Engineering & Technology**

*Website: www.sciencepubco.com/index.php/IJET*

***Research paper***

# Nature inspired algorithm using particle swarm approach with variations in inertia weights for automatic test data generation based on dominance concepts

**Sanjay Singla [1] *, Raj Kumar [2], Dharminder Kumar [3]**

[1] *Research Scholar, Department of Computer Science & Engineering, University Institute of Engineering and Technology, Maharishi Dayanand University, Rohtak, Haryana, India*
[2] *Department of Computer Science & Engineering, University Institute of Engineering and Technology, Maharishi Dayanand University, Rohtak, Haryana, India*
[3] *Department of Computer Science & Engineering, Guru Jambheshwar University, Hisar, Haryana, India*

**Abstract**

In software testing, testing of all program statements is a very crucial issue as it consumes a lot of time, effort and cost. The time, effort and cost can be reduced by using an efficient technique to reduce the test case and a good optimization algorithm to generate efficient, reliable and unique test cases. In this paper, the concept of dominance tree is used which covers all edges/statement by using minimum test case. Nature inspired algorithm - PSO (Particle Swarm Optimization) by applying different inertia weights is used to generate unique, reliable and efficient test cases to cover the leaf nodes of dominance tree. Inertia weights like fixed inertia weight (FIW), global-local best (GLbestIW), Time-Dependent weight (TDW), and proposed GLbestRandIW weights are used with PSO to investigate the effect of inertia weights on the execution of PSO with respect to number of generation required, percentage coverage , total test cases generated to test the software under consideration.

***Keywords***: *Testing; Particle Swarm Optimization (PSO); Inertia Weight; Dominance Tree.*

## 1. Introduction

In software testing, the complete or exhaustive testing is not possible. As testing is most difficult and time-consuming process. A major portion of the cost involved in software development life cycle is due to testing. Testing of each and every statement of the program is crucial. Instead of checking each and every statement is coved or not, a technique called dominance tree that covers almost each and every statement. Dominance tree covers provide the leaf nodes that have to be tested, now concentrating only on these leaf nodes, almost each and every statement can be covered. Thus instead of generating test cases for each and every statement, the test case is generated to test leaf nodes of dominance tree. Thus, these will reduce efforts involved in testing to large extent. The dominance tree concept is explained is explained in a further section.

Selection of test data is most difficult/ critical task [17] and it requires very good skill in test data selection/generation method. Many researchers have proposed their own different techniques on their own time to create good investigation statistics during software testing [5]-[8], [10]-[13], [19]. Effort, cost along with the time can be minimizing if the process of testing becomes automatic. In last 20 years, there are a lot of optimization techniques were introduced which proves themselves in the area of software testing [16], [17] but processing time to explore and exploit the promising reasons in the solution domain is affected.

Natural selection is also the basis of Genetic algorithm. The concept was given by Darwin in 1960[1]. GA uses the population of chromosomes and undergoes various operations like mutation, crossover to produce a new generation [2]. GA proves itself in a number of engineering and optimization problems [3]. In some cases, it is unable to deal with the problems of local minima and local maxima. Furthermore, all this is took a large amount of execution time. On the other end, PSO in light of the social conduct of birds flocking [14]. In the year 1995 Kennedy and Eberhart anticipated PSO. The concept of particle best and global best introduce the memory concept in PSO and make this algorithm more fast and better as compared to GA [15],[20]. Owing to its unfussiness, greater convergence characteristics as well as high precision, PSO proves its effectiveness for complex optimization problems. This paper manages a productive PSO algorithm for software testing and examined the effects of weight of inertia variations. The proposed technique gives the outcomes better percentage coverage and less number of generations.

## 2. Background

This section explains the fundamental concepts, which further will be used in the considered problem domain.

Control flow graph (CFG)

CFG is a diagrammatic representation which itself is able to explain the flow of a program. It is a directional graph. It is represented as:

$G = \{V, E\}$

Where, "G" represents a Graph with directions (CFG). "V" is set, which represents the vertices or nodes of "G". In actual, "V" reflect the Instructions in the programme or unique steps of the programme. "E" represents the edges of the directed graph. This is again a set. It reflects the movement of the programme pointer during the execution of the programme. Figure1, demonstrates the instruction set whose directed graph (CFG) is represented with the help of Figure2. Figure 4 shows the CFG of greatest of three numbers program.

Dominance tree

A dominance tree (DT) is a directed tree of a graph G={V,E}. This tree must full fill some properties. Firstly, each vertex $V_i$ of G other than the root of G, must be ahead of a single edge every time. Another property is the existence of dominance path for each vertex of $V_i$ from its root. Here the dominance path reflects the directed path from root node of the graph G to every other vertex of the graph and follows the criteria that a vertex Vi dominate other vertex Vj, if and only if the path from root node to Vj always contains Vi and $i \neq j$ [17]. Figure3 reflects the dominance path of the code represented by Figure1 and control flow graph shown in Figure4. Figure4 and Figure5 show the control flow graph and dominance tree of Greatest of three number program. The Path_Dominance (7) = [1, 2, 7] represents the dominance path of vertex 7 of Figure3.

```
0 1 program test1;
1 1 variable x, y, z: int;
2   1   Start:
3   1   x: = 0;
4   1   y: = 0;
5   1   read (z);
6   2   while (z <> 0)
7   2   Start
8   3   if (z mod 2) = = 0
9   4   then x: = x + 1
10  5   else
11  5   y: = y +1;
12  6   read (z)
13  6   end;
14  7   write(x, y);
15  7   end.
```

**Fig. 1:** Code1.

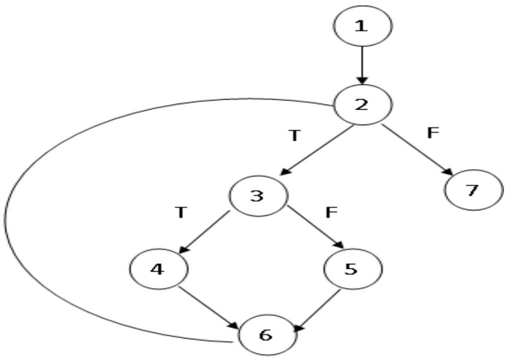

**Fig. 2:** CFG of Code1.

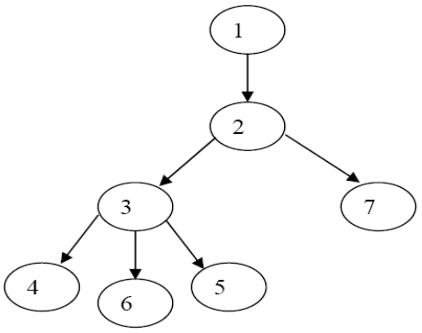

**Fig. 3:** Dominance Tree.

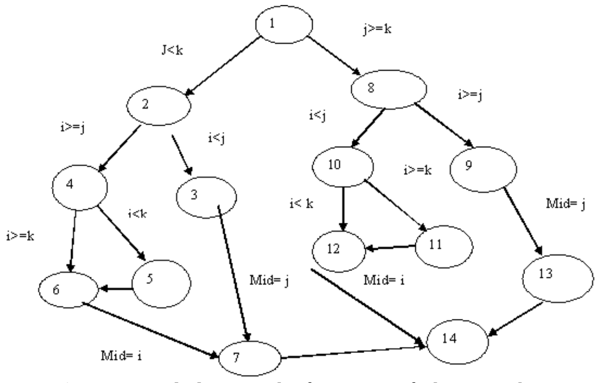

**Fig. 4:** Control Flow Graph of Greatest Of Three Number.

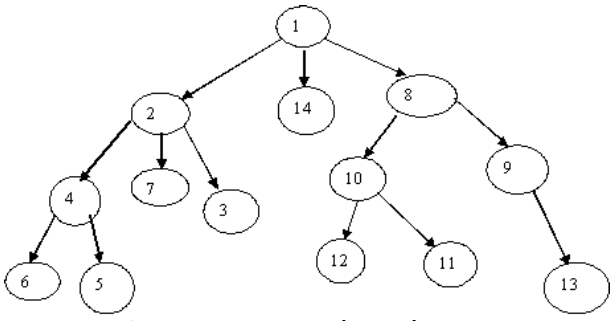

**Fig. 5:** Dominance Tree of CFG of Figure 4.

Test cases minimization

The target of the study is the testing of entire group of statements of the program by using minimum test cases selection. This may be achieved by the selection of lesser number of nodes/statements that ensures the scope of every statement resides in the program [18]. The principle concern is all leaves nodes of the dominance graph. It is understandable that set of the paths which cover these leaves must also covers the entire nodes in the tree. Leave vertices of Figure3 are represented as L= [4, 5, 6 and 7]. The dominance path can be expressed as:

Path_dominance (4) = [1-2-3-4]
Path_dominance (6) = [1-2-3-6]
Path_dominance (5) = [1-2-3-5]
Path_dominance (7) = [1-2-7]

It is observed that all the vertices of CFG represented by Figure2 are covered by performing dominance path analysis of leave nodes. This coverage of every vertex of the graph also represents the coverage of every single statement of the code, which are the major criteria behind the software testing.

## 3. Particle swarm optimization (PSO)

Kennedy and Eberhart developed an algorithm called Particle Swarm Optimization [14] that replicates the natural tendency of birds or fishes during their food discovery or new appropriate habitation. Consider a d-dimensional search space in the basic PSO technique.

1) Every member is considered as a particle. Each particle is shown by d-dimensional positional vector along with description as $X_i = [x_{i1}, x_{i2}, \ldots\ldots, x_{id}]$
2) A Population is an ordered set of particles in the swarm which is expressed as pop=[$x_1$, $x_2$,...,$x_d$].
3) pBest is considered as the previously best value of every particle. This is expressed as $PB_i = [pb_{i1}, pb_{i2}, \ldots\ldots, pb_{id}]$
4) gBest is considered as global best for each particle and can be calculated as. $GB_i = [gb_{i1}, gb_{i2}, \ldots\ldots, gb_{id}]$
5) The term Velocity defines the change in the position of each particle and is expressed as: $V_i = [v_{i1}, v_{i2}, \ldots\ldots, v_{id}]$

When number of iterations is "k" then the velocity of ith particle is expressed as:

$$v_{id}(k+1)=wv_{id}(k)+c_1r_1(pb_{id}(k)-x_{id}(k))+c_2r_2(gb_{id}(k)-x_{id}(k)) \quad (1)$$

Where i varies from 1 to n. Here, n is the size of each population, inertia weight is denoted by w, $c_1$ along with $c_2$ are constants. $r_1$ and $r_2$ represents random variables having scope [0,1].

6) The position of particles are expressed with the help of following equation:

$$x_{id}(k+1)=x_{id}(k)+v_{id}(k+1) \quad (2)$$

Figure 6 represents the programme flow of proposed algorithm.
Considerations for inertia weight
Fixed inertia weight (FIW)
The usual PSO algorithm at first utilized a steady or constant inertia weight.
Time dependent weights (TDW)
Keeping in mind the end goal to enhance the existing system, the time-differing inertia weight was recommended [4]. This inertia weight directly diminishes regarding time. For the most part, for starting phases of the pursuit procedure, large inertia weight to upgrade the global exploration (looking new region) is prescribed while, for end part, low inertia weight is proposed for local investigation.
Global-Local best inertia weight (GLbestIW)
The GLbestIW strategy is projected in [9]. It is considered as a function of local and global estimations of the particles in various generations. The equation for the same is given as:

$$\text{GLbestRandIW Wi} = \left(1.1 - \left(\frac{\text{gbesti}}{\text{pbesti}}\right)\right)$$

Proposed global-local best random inertia weight (GLbestRandIW)
The changes in inertia factors may enhance the performance of this optimization technique (PSO). Here, inertia weight is considered as a function of pbest and gbest with random factor values of the particles in each generation.

$$\text{GLbestRandIW Wi} = \left(1.1 - \left(\frac{\text{gbesti}}{\text{pbesti}}\right)\right) * z(\text{Rand}) + 0.5 * (\text{Rand})$$

Where Z= 4* (Rand) *(1-Rand)

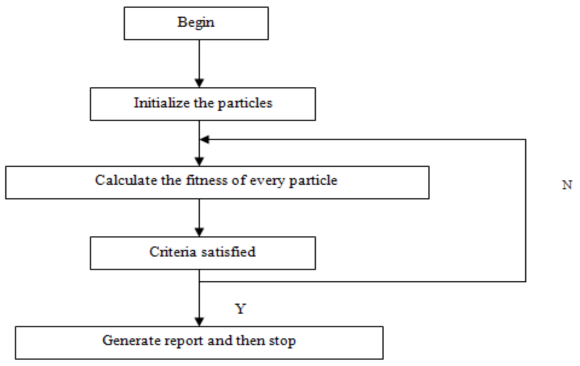
Begin
Initialize the particles
Calculate the fitness of every particle
Criteria satisfied
N
Y
Generate report and then stop

**Fig. 6:** Flow Chart of PSO.

## 4. Fitness function

All algorithms in this manuscript utilized a fitness function that used the dominance relations ideas between nodes of control flow graph of the program. This fitness function represents the ratio of the number of covered nodes under dominance path analysis of the node under consideration to the total number of dominance path nodes. During the execution of the program, every test case is applied and results are observed under execPath. The computation of fitness value ft(Vi) by considering isolated or unique chromosome $v_i$ (i=1… S) In a population of size 'S' is performed as:

1) Locate a set of vertices enclosed by a test scenario: find out execPath
2) verify the dominance path of the node under consideration $\text{Path}_{\text{Dominance}}(n)$
3) Identify nodes which are not covered: discover ($\text{Path}_{\text{Dominance}}(n)$ - execPath)
4) Identify already covered nodes: discover ($\text{Path}_{\text{Dominance}}(n)$ - execPath)'
5) Count number of covered nodes : count |($\text{Path}_{\text{Dominance}}(n)$ - execPath)'|

At that point

$$\text{ft(vi)} = \frac{|(\text{Path}_{\text{Dominance}}(n) - \text{execpath})'|}{|\text{Path\_Dominance}(n)|}$$

Particle or the individual is represented by test case. The fitness value 1 i.e. $ft(v_i)$ = 1 for a test case $v_i$ then this test case is optimal [18]. The solitary method for optimization algorithms using feedback is fitness value.

## 5. Experimental results and conclusion

Trials are performed on normally utilized programs as shown in table 1. PSO with variations in weights of inertia, a fixed inertia weight (FIW), Time-Dependent weight (TDW) and global-local best inertia weight (GLbestIW) and global-local best random inertia weight (GLbestRandIW) is used to generate test cases is evaluated using dominance tree concepts. The different methods are shown in Table 2.

**Table 1:** Program List

| Name of Program |
|---|
| LTV (Largest among Three values) |
| PNG (Prime Number Generation) |
| RF (Remainder Function) |
| ROI (Rate of Interest) |
| PQE (Program for Quad. Equation) |
| AFT (Triangle's Area) |
| HCF (Highest Common Factor) |

**Table 2:** PSO Methods with Different Inertia Weight

| PSO method description | Method name |
|---|---|
| FIW | M1 |
| TDW | M2 |
| GLbestIW | M3 |
| GLbestRandIW | M4 |

In table 3, comparison between M1, M2, M3 and M4 is shown in the terms of number of generation required to cover the test cases. The M4 takes less number of generations as compare to other methods. The number of generation required to complete testing process is directly indicates the time taken by the algorithm to process. Therefore, M4 is much faster as compare to other three methods. Hence performance of M4 is best among other three methods. The graph in Fig 7 shows the comparison of different methods with respect to number of generation each method required to complete the testing process.

**Table 3:** Comparison in Terms of Number of Generations

| Prog No | M1 | M2 | M3 | M4 |
|---|---|---|---|---|
| 1 | 15 | 14 | 11 | 9 |
| 2 | 4 | 3 | 3 | 2 |
| 3 | 9 | 7 | 5 | 3 |
| 4 | 5 | 5 | 4 | 2 |
| 5 | 11 | 9 | 7 | 6 |
| 6 | 16 | 14 | 10 | 9 |
| 7 | 6 | 5 | 5 | 4 |

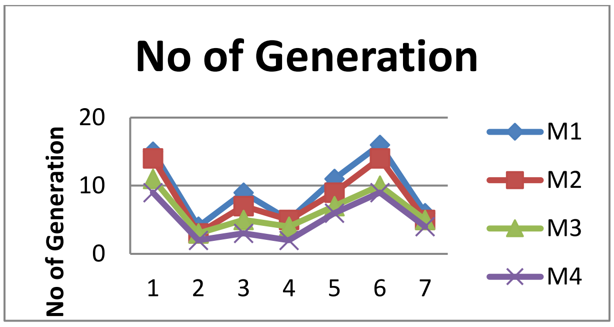

**Fig. 7:** Evaluation Focusing Number of Generations.

Table 4 shows the comparison of percentage coverage ratio of all four methods. All programs are 100% covered by method M4. Hence M4 is better than all three method as its percentage coverage ratio is high than others in all program, the same is also shown in fig 8.

**Table 4:** CRP Comparison

| Prog No | M1 | M2 | M3 | M4 |
|---|---|---|---|---|
| 1 | 100 | 100 | 100 | 100 |
| 2 | 100 | 100 | 100 | 100 |
| 3 | 100 | 100 | 100 | 100 |
| 4 | 100 | 100 | 100 | 100 |
| 5 | 100 | 100 | 100 | 100 |
| 6 | 94 | 98 | 100 | 100 |
| 7 | 91 | 97 | 99 | 100 |

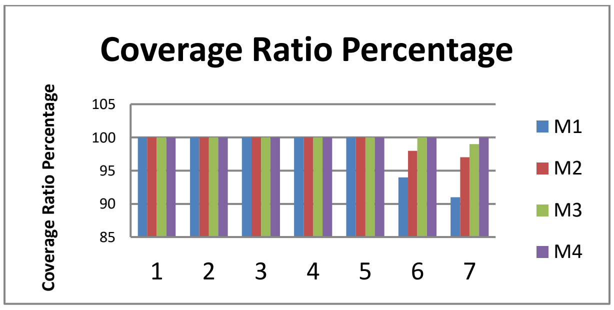

**Fig. 8:** Assessment with Respect to Coverage Ratio Percentage.

By study of fig 9 and table 5, the quantity of test cases produced in M4 is less in compared to other three PSO methods. M4 shows entire coverage by using less number of iterations as compared to others, which represents that, test cases produced by M4 are unique as compared to other methods. Further, it can be concluded that M4 performs better as compared to other methods.

**Table 5:** Effect of Total Test Cases

| Prog no | M1 | M2 | M3 | M4 |
|---|---|---|---|---|
| 1 | 105 | 98 | 77 | 63 |
| 2 | 40 | 30 | 30 | 20 |
| 3 | 81 | 63 | 45 | 27 |
| 4 | 25 | 25 | 20 | 10 |
| 5 | 88 | 72 | 56 | 48 |
| 6 | 160 | 140 | 100 | 90 |
| 7 | 54 | 45 | 45 | 36 |

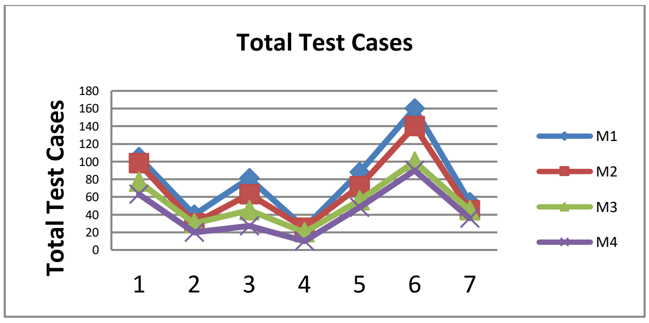

**Fig. 9:** Evaluation with Respect to Number of Test Cases Generated.

## References

[1] Girgis MR, "Automatic test data generation for data flow testing using genetic algorithm", *Journal of Universal Computer Science*, Vol.11, No.6, (2005), pp.898–915.

[2] Pargas RP, Harrold MJ & Peck RR, "Test Data Generation using Genetic Algorithms", *Software Testing Verification and Reliability*, Vol.9, (1999), pp.263-282. https://doi.org/10.1002/(SICI)1099-1689(199912)9:4<263::AID-STVR190>3.0.CO;2-Y.

[3] Alander JT, Mantere T & Turunen P, "Genetic Algorithm Based Software Testing", *Proceedings of International Conference*, (1998), pp.325-328. https://doi.org/10.1007/978-3-7091-6492-1_71.

[4] Abido MA, "Multiobjective particle swarm optimization technique for environmental/economic dispatch problem", *Electric Power System Research*, Vol.79, No.7, (2009), pp.1105–1113. https://doi.org/10.1016/j.epsr.2009.02.005.

[5] Boyer R, Elspas B & Levitt K, "Select-a formal system for testing and debugging programs by symbolic execution", *SIGPLAN Otices*, Vol.10, No.6, (1975), pp.234-245. https://doi.org/10.1145/390016.808445.

[6] Clarke L, "A system to generate test data and symbolically execute programs", *IEEE Transaction on Software Eng.*, Vol.SE-2, No.3, (1976), pp.215- 222. https://doi.org/10.1109/TSE.1976.233817.

[7] Ramamoorthy C, Ho S & Chen W, "On the automated generation of program test data", *IEEE Trans. Software Eng*., Vol.SE-2, No.4. (1976), pp.293-300. https://doi.org/10.1109/TSE.1976.233835.

[8] Howden W, "Symbolic testing and the DISSECT symbolic evaluation system", *IEEE Trans. Software Eng*., Vol.SE-4, No.4, (1977), pp.266- 278. https://doi.org/10.1109/TSE.1977.231144.

[9] Arumugam MS & Rao MVC, "On the performance of the particle swarm optimization algorithm with various inertia weight variants for computing optimal control of a class of hybrid systems", *Discrete Dynamics in Nature and Society*, (2006). https://doi.org/10.1155/DDNS/2006/79295.

[10] Ince D, "The automatic generation of test data", *Computer Journal*, Vol.30, No.1, (1987), pp.63-69. https://doi.org/10.1093/comjnl/30.1.63.

[11] Miller W & Spooner D, "Automatic generation of floating-point test data", *IEEE Trans. Software Eng.*, Vol.SE-2, No.3, (1976), pp.223-226. https://doi.org/10.1109/TSE.1976.233818.

[12] Offutt J, Jin Z & Pan J, "The Dynamic domain reduction procedure for test data generation", *Software Practice and Experience*, Vol.29, No.2, (1997), pp.167–193. https://doi.org/10.1002/(SICI)1097-024X(199902)29:2<167::AID-SPE225>3.0.CO;2-V.

[13] Gupta N, Mathur AP & Soffa ML, "Automat geneticed test data generation using an iterative relaxation method", *ACM SIGSOFT Sixth International Symposium on Foundations of Software Engineering*, (1998), pp.231–244.

[14] Kennedy J & Eberhart R, "Particle swarm optimization", *IEEE International Conference on Neural Networks*, (1995), pp.1942–1948. https://doi.org/10.1109/ICNN.1995.488968.

[15] Narmada N & Mohapatra DP, "Automatic Test Data Generation for data flow testing using Particle Swarm Optimization", *Communications in Computer and Information Science*, Vol.95, No.1, (2010), pp.1-12.

[16] Michael CC, McGraw GE & Schatz MA, "Generating software test data by evolution", *IEEE Transactions on Software Engineering*, Vol.27, No.12, (2001), pp.1085-1110. https://doi.org/10.1109/32.988709.

[17] Ghiduk AS, Harrold MJ & Girgis MR, "Using Genetic Algorithms to Aid Test-Data Generation for Data-Flow Coverage", *14th Asia-Pacific Software Engineering Conference*, (2007). https://doi.org/10.1109/ASPEC.2007.73.

[18] Ghiduk AS & Girgis MR, "Using Genetic Algorithms and dominance concepts for generating reduced test data", *Informatics*, Vol.34, (2010), pp.377-385.

[19] Chang KH, Cross JH, Carlisle WH & Brown DB, "A framework for intelligent test data generation", *Journal of Intelligent and Robotic Systems-Theory and Application*, Vo.5, No.2, (1992), pp.147-165. https://doi.org/10.1007/BF00444293.

[20] Biswas A, Mishra KK, Tiwari S & Misra AK, "Physics-inspired optimization algorithms: a survey", *Journal of Optimization*, (2013).