STM JOURNALS

Journal of Artificial Intelligence Research & Advances
ISSN: 2395-6720(online)
Volume 2, Issue 2
www.stmjournals.com

Applying Computational Intelligence in Software Testing

Saumya Dixit*, Pradeep Tomar
School of Information and Communication Technology, Gautam Buddha University, Greater Noida, India

Abstract

This paper describes two computational techniques that can be used to automate generation of test cases. These techniques are Genetic Algorithm (GA) and Particle Swarm Optimization (PSO). Both the algorithms are evaluated by varying parameters and recording the change in the results. A comparative study based on the results of these algorithms is also done.

Keywords: *Computational techniques, genetic algorithm (GA), particle swarm optimization (PSO), software, algorithms*

**Author for Correspondence* E-mail: saumyadixit0007@gmail.com

INTRODUCTION

High quality of software is must in these times when science and technology is at its peak. Software is used everywhere these days. The only way to deliver a high quality of software is to perform high quality of software testing. Software testing is the task that can ensure more reliable software.

During the development of software testing activities are not usually carried out in a proper manner which leads to a poor quality software. There is a need to perform the testing activities in a well specified and organized manner. Even, if the testing activities are carried out during the software development in an organization then the amount of time it consumes is very high, if it is conducted properly. It takes about 40–60 percent of the total development process [1–4]. There are two types of testing: manual testing and automated testing.

The manual testing process is very time consuming. Manual testing is the process of manually testing software for defects. It requires a tester to play the role of an end user and use most of all features of the application to ensure correct behavior. Manually repeating these tests is costly and time consuming. Once created, automated tests can be run over and over again at no additional cost and they are much faster than manual tests. Automated software testing can reduce the time to run repetitive tests from days to hours. Automating testing process reduces time as compared to manual testing, is more reliable, more comprehensive and more reusable. This paper mentions two such computational intelligence techniques i.e. GA and PSO. Both of these are used to generate test cases separately and evaluation is done by changing various parameters of these algorithms.

GENETIC ALGORITHM

Testing ensures that software meets user specifications and requirements. However, the field of software testing has a number of underlying issues like effective generation of test cases which need to be tackled. These issues demand on effort, time and cost of the testing.

Different techniques and methodologies have been proposed for taking care of these issues. Use of evolutionary algorithms for automatic test generation has been an area of interest for many researchers. Genetic Algorithm (GA) is one such form of evolutionary algorithms [2–6].

It can be explained by the following pseudo code:

```
Initialize (population)
Evaluate Fitness of the each individual
While (termination condition is met) do
{
Selection (of parents from population)
Cross-over (of selected parents to produce offspring)
```


Mutation (of the offspring and parents)
Evaluate (the new population consisting of offspring and their parents)
}

It implements biological evolution phenomenon of survival of the fittest and reproduction through cross breeding [1, 7–9]. The principle behind GA is that it creates and maintains a population of individuals represented by chromosomes.

By crossover and mutation on chromosomes, the population evolves from one generation to another until meeting the termination conditions. The genetic operations in every GA include selection, recombination and mutation and have been well studied in many literatures.

The start of the evolution takes place from a population of randomly generated individuals. The evolution happens in generations. A fitness function is predetermined in order to evaluate every individual. The individuals are selected based on their fitness from the current population. These individuals are then modified to form a new generation.

The termination conditions that are usually encountered are either maximum number of generations has been reached or the desired level of the fitness has been achieved. If the termination is due to the maximum number of generations then the satisfactory solution may or may not be achieved.

PARTICLE SWARM OPTIMIZATION

PSO Algorithm is another evolutionary technique which is based on movement and intelligence of swarms. It uses a number of agents (particles) that constitute a swarm moving around in the search space looking for the best solution. Every particle is considered as a point in N-dimensional space.

The particle adjusts its flying according to its own flying experience as well as others' flying experience. It can be explained by the following steps which are repeated until stopping conditions are met:

- Initialize the population of individuals with current position and velocity.
- Evaluate the current fitness of each individual particle (Pbest).
- Keep track of the highest fitness (Gbest) among all the individuals.
- Update the velocity based on Pbest and Gbest location.
- Modify the position of the particle accordingly.

Aloka *et al*. describe PSO as an optimization technique based on social behavior of bird flocking or fish schooling [3, 10]. PSO has many similarities with evolutionary computation techniques like GA. The system is initialized with a population of random solutions and searches for optima by updating generations. Each particle has a position and a non-zero velocity at any instant and is aware of the global best and the local best (self-best) positions.

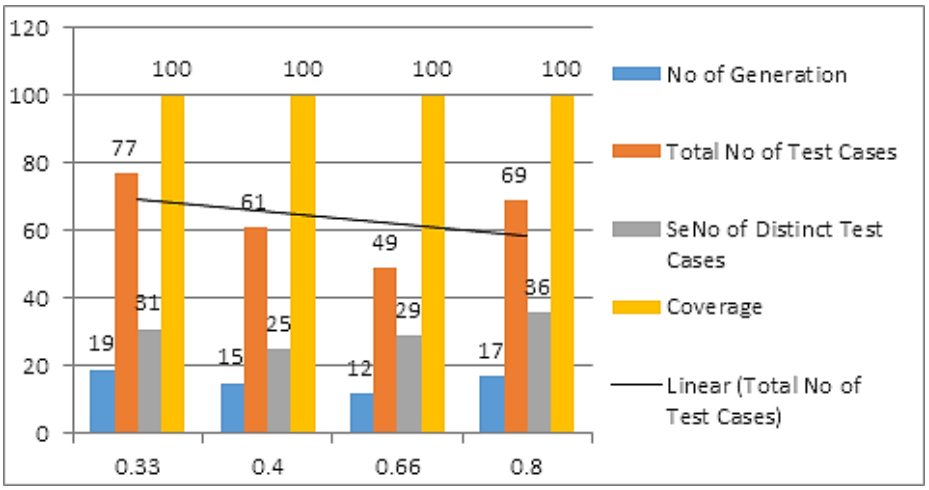
They fly through the problem space by following the current global and local best and approach towards optimality. Some of the attractive features of the PSO include ease of implementation and the fact that no gradient information is required. In PSO, the solution space of the problem is formulated as a search space. Each position in the search space is a potential solution of the problem. Particles cooperate to find the best position (best solution) in the search space (solution space). Each particle moves according to its velocity.

IMPLEMENTATION

GA and PSO are implemented and their results are recorded. GA and PSO are implemented using Microsoft Visual Studio 2010 in C# programming language. A set of programs are given as input to these algorithms and for these programs the test cases are generated. The parameters include population size, length of chromosomes and maximum number of generations, probabilities of the crossover and mutation operators in case of GA. The parameters in case of PSO are like number of agents, maximum iteration, dimensions.

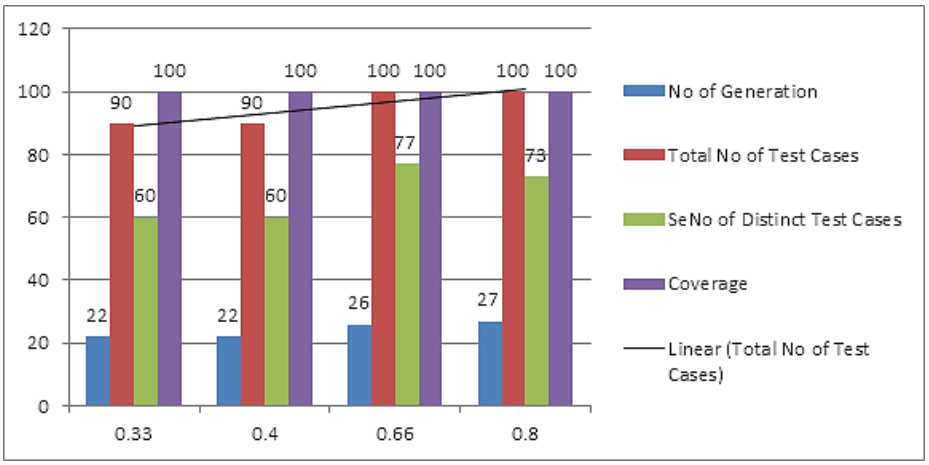
In Figure 1 results are obtained by keeping the mutation rate constant as 0.05 and changing the value of crossover probability as 0.33, 0.4, 0.66 and 0.8 and taking the chromosome of 12 length.

In Figure 2 results are obtained by keeping the mutation rate constant as 0.05 and changing the value of crossover probability (0.33, 0.4, 0.66 and 0.8) and taking the chromosome of 18 length. Figure 3 depicts that increase in the length of chromosomes increases the number of generations. In Figure 4 effect of varying the Number of agents as 2, 4 and 6 are recorded when dimension is kept as 3 and maximum number of iterations is 40.



Crossover Probability

Fig. 1: *Results of GA with Varying Cross Probability.*



Crossover Probability

Fig. 2: *Results of GA with Varying Cross Probability.*

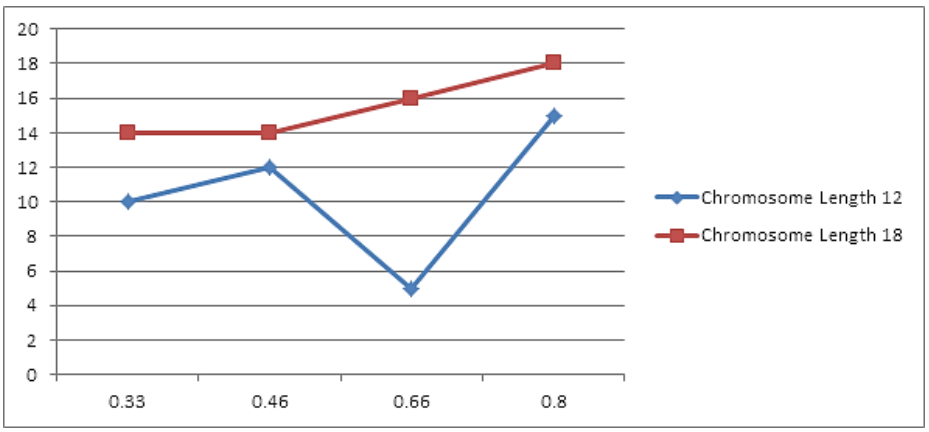


Fig. 3: *Results of GA with Varying Chromosome Length.*

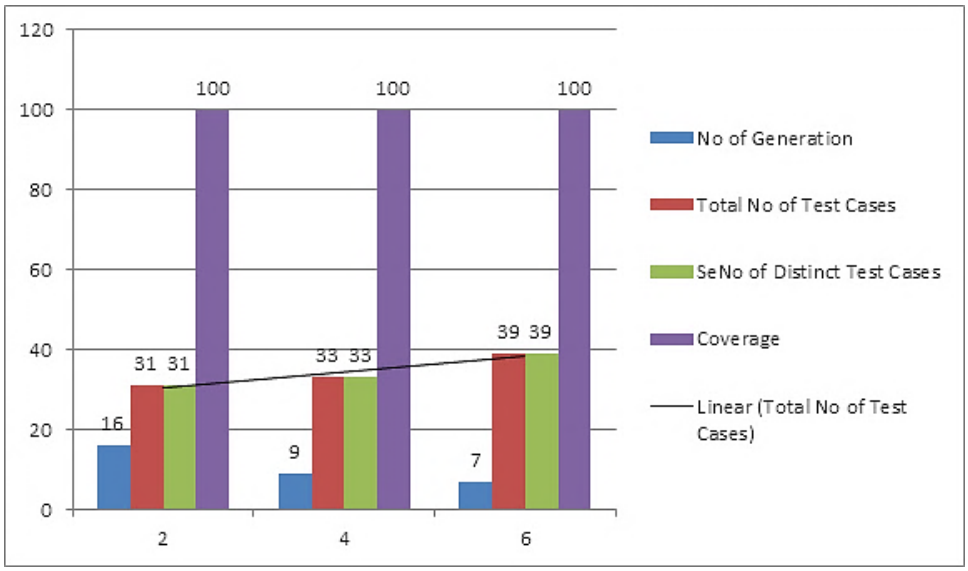


Fig. 4: *Results of PSO with Varying Number of Agents.*

It is understood that in PSO with the increase in number of agents iteration decreases. It is further noticed that in PSO number of distinct test cases generated is almost equivalent to the number of test cases generated. Therefore, PSO produces distinct test cases.

CONCLUSION

It was found that PSO is better than GA in terms of number of test cases. Since PSO keeps track of global best position it generated more number of distinct test cases which proves to be more efficient in software testing. PSO also has very few parameters in comparison to GA. But it was found that PSO lacked the phenomenon of mutation and crossover that helps to introduce new genes in the chromosome. GA has a parallel nature of search and can solve both discrete and continuous problems whereas PSO can solve only discrete problems. PSO is easy to implement as compared to GA but it is problem dependent as it is very difficult to implement when it comes to non-coordinate systems.

Though, both the algorithms are population based evolutionary techniques they both have their pros and cons. The paper described the concepts of GA and PSO and the results obtained on their implementation and evaluation by changing the initial parameters taken.

This approach is intended to be extended in order to develop a hybrid algorithm that combines the power of both the algorithm and hence can be used in the software testing.

REFERENCES

1. Kewen Li, Zilu Zhang, Jisong Kou. Breeding software data with genetic particle swarm mixed algorithm. *Journal of Computers*. 2010; 5(2): 258–265p.
2. Chayanika Sharma, Sangeeta Sabharwal, Ritu Sibal. A survey on software testing techniques using genetic algorithm. *International Journal of Computer Science Issues*. 2013; 10(1): 13p. ISSN: 1694-0784.
3. Aloka S, Peenu Singh, Geetanjali Rakshit, et al. Test effort estimation-particle swarm optimization based approach. *6th Mexican International Conference on Computer Science (ENC 2005)*. 2011; 168: 463–474p.
4. Abdelaziz M Khamis, Moheb R Girgis, Ahmed S Ghiduk. Automatic software test data generation for spanning sets coverage using genetic algorithms. *Computing and Informatics*. 2007; 26(4): 383–401p.
5. Ahmed AA Esmin, Stan Matwin. HPSOM: A hybrid particle swarm optimization algorithm with genetic mutation. *International Journal of Innovative Computing, Information and Control*. 2013; 9(5): 1919–1934p.

6. Peng Nie. A PSO test case generation algorithm with enhanced exploration ability. *Journal of Computational Information Systems.* 2012; 8(14): 5785–5793p.
7. Girgis MR. Automatic test data generation for data flow testing using genetic algorithm. *Journal of University Computer Science*. 2005; 11(6): 898–915p.
8. Zhong Wen Liang, Wang Hui Sen, Zhang Jun, et al. Novel particle swarm optimization with heuristic mutation. *Computer Engineering and Design*. 2008; 3402–3405p.
9. Liu Shou Sheng, Yu Sheng Lin, Ding Yong, et al. Multipopulation parallel genetic algorithm based on even partition. *Journal of Data Acquisition & Processing*. 2003; 142 –145p.
10. Marr´e M Bertolino. Using spanning sets for coverage testing. *IEEE Transactions on Software Engineering*. 2003; 29(11): 974–984p.

Cite this Article
Saumya Dixit, Pradeep Tomar. Applying Computational Intelligence in Software Testing. *Journal of Artificial Intelligence Research & Advances*. 2015; 2(2): 7–11p.