A Systematic Review of Software Testing Using Evolutionary Techniques

Deepti Bala Mishra[1], Rajashree Mishra[2(✉)], Kedar Nath Das[3], and Arup Abhinna Acharya[1]

[1] School of Computer Engineering, KIIT University, Bhubaneswar 751024, India
dbm2980@gmail.com, aacharyafcs@kiit.ac.in
[2] School of Applied Sciences, KIIT University, Bhubaneswar 751024, India
rajashreemishra011@gmail.com
[3] Department of Mathematics, NIT Silchar, Assam, India
kedar.iitr@gmail.com

Abstract. A best solution for decreasing software cost and reducing the cycle time during software development is automatic software testing and it has been seen by various organization. User specifications and requirements can be fully achieved by software testing. A number of issues are underlying in the field of software testing such as prioritization of test cases and automatic and effective test case generation are to be handled properly and they mostly depends on duration, cost and effort during the testing process. Testing can be done in two different ways such as manual testing and automatic testing by using different testing tools. Manual testing are very time consuming and this can be overcome by automatic testing by generating test cases automatically. Several types of evolutionary techniques like Genetic Algorithm, Particle Swarm Optimization and Bee Colony Optimization have been used for software testing. In this research paper, a survey of different evolutionary techniques used in software testing have been presented by taking the various issues in to account.

Keywords: Test data generation · Software testing · Genetic algorithm (GA) · Particle swarm optimization (PSO) · Bee Colony Optimization (BCO)

1 Introduction

Now-a-days automated software testing and developing of high quality test cases are two main objectives in the software industry. To support a high quality assurance of software, to create reliable, robust and trust worthy software or to deliver error free software, testing is performed by gathering required information of the software. It is also defined by the process of verification and validation, which meets the technical and business requirements [1, 2] in software development process. Testing is a most time consuming task which takes approximately 60% work load of the total software development time. If the testing is performed using automated testing then it will lead to reduce in software development cost by a significant margin [3–5].

K. Deep et al. (eds.), *Proceedings of Sixth International Conference on Soft Computing for Problem Solving*, Advances in Intelligent Systems and Computing 546, DOI 10.1007/978-981-10-3322-3_16

A best solution for decreasing software cost and reducing the cycle time during software development is automatic software testing and it has been seen by various organization [6]. By using different software tools software can tested either manually or automatically. It is proved that automated software testing is better than manual testing as manual testing is a very time consuming and expensive task [7, 8]. Various types of techniques have been proposed by researchers and a lot of work has been done for software testing using soft computing techniques such as GA, Neural Network, genetic programming, fuzzy logic and evolutionary computing by providing high quality test data [8–10]. These techniques can be applied for test data generation to optimized problems.

This paper presents a survey of how different types of evolutionary techniques such as GA, PSO and BCO have been efficiently used in software testing and have been applied extensively for automated test data generation. Further the paper is partitioned into 4 sections. Section 1 presents the Introduction to software testing, Sect. 2 presents related work in the field of software testing using different types of evolutionary techniques, Sect. 3 contains a brief description about the working of GA, PSO and BCO and Sect. 4 gives a conclusion followed by our future work.

2 Related Work

This section provides a survey on different evolutionary techniques like GA, PSO and BCO used in software testing field for generating best test cases.

Last et al. [11], proposed a hybrid fuzzy based GA, which is an age extension of GA (FAexGA) to generate test cases for mutation testing. They found a very minimal set of test cases. The faults in test cases are exposed by the use of mutated versions of the original method. The proposed method uses a FLC (Fuzzy Logic Controller) for obtaining the probability of crossover. The probability of crossover differs according to the age intervals allocated during lifetime. The life time and age of chromosomes (parents) are defined by the FLC state variables. The truth value for obtaining Young-age, middle-age and old-age are shown in Table 1. Where,

Table 1. Fuzzy rule for cross over probability [14]

	Parent 1			
		Young-age	Middle-age	Old-age
Parent 2	Young-age	Low	Medium	High
	Middle-age	Medium	High	Medium
	Old-age	Low	Medium	Low

Age ∈ [Young-age, Middle-age, Old-age]
Crossover Probability ∈ [Low, Medium, High]

In their work an effective set of test cases are generated for a Boolean expression of 100 Boolean attributes by using three logical operators AND, OR, and NOT. An external application generates the correct expression randomly and one simple function

is evaluated for each test case to generate an erroneous expression. Here a 100-bit lengthen binary strings of one dimension are generated as chromosomes.

Hla et al. [12], proposed a particle swarm optimization (PSO) algorithm based on modified software units for embedded real time software regression testing. The proposed algorithm prioritize the test cases automatically so that new higher priority test cases are selected for regression testing. The PSO algorithm successfully applied to the prioritization problem by taking solution as particle space and from which the best new positions of test cases, based on software unit can be found. Their results shows that the PSO algorithm can prioritized the test cases in the test suites by new best positions effectively and efficiently.

McCaffrey [13], developed a simulated BCO algorithm by which pair wise test sets can be generated to reduce the test set size as all systems are not supported for exhaustive testing with all possible inputs. The technique is a combinatorial NP-hard technique and it takes more time to generate test sets, which are far better than the test sets generated by deterministic approach.

Nachiyappan et al. [14], proposed a model based on genetic algorithm to decrease the cost of regression testing. Their proposed model creates population by taking the test history, the fitness value is calculated depending on the block based coverage value and run time of test case and the genetic operators are used for successive generations till the test cases with optimum value is found. They used Average Percentage of Faults Detected (APFD) metric to calculate the fitness function of individual test cases. The APFD can determine the effectiveness about cost, coverage, runtime and ordering of the new test case. The test cases are rejected which violates the specified time constraints. The model shows a good optimal sized test set by reduced test suite technique and the method is very highly adaptive as test case reduction is more effective when the fitness granularity is increased.

Kaur and Goyal [15], presented a BCO algorithm for fault coverage to a maximum limit. The authors have mapped the farmer bee's scenarios to prioritize the test suite. They explained their work by taking two examples like "college program for admission in courses" and "Hotel Reservation". In their work values have been compared using APFD (Average Percentage of Fault Detection) metrics and the proposed algorithm has been implemented in CPP compiler.

Ferrer et al. [16], presented two search based approaches as GTSG (Genetic Test Sequence Generator) and ACOts (Ant Colony Optimization approach for Test Sequence) for test sequence generation in functional testing with shortest valid path, which covers full transition and class. They used one CIT (Combinatorial Interaction Testing) approach, the classification tree method for test planning and design in functional testing. The authors defined the entire model as an extended classification tree to generate test sequences for a SUT (Software Under Test), which is needed for both industry and academia. Their first approach is GTSG with memory operator to preserve the memory for population evaluation as well as faster computation to get the solution. The second one is ACOts, which deals with large construction graphs. Test sequence can be generated with near-optimal solutions, where search spaces are separated. The authors performed the experiments using 12 software models by comparing their proposed approaches with greedy algorithm and they found their approaches can

generate test sequences with shortest valid path, which covers full transition and class in functional testing.

Ankur and Srivastav [17], used GA to generate test data automatically for branch testing. They developed an improved approach which focuses on branch ordering, memory and elitism. The authors have discussed about DFS (Depth First Strategy), BFS (Breadth First Strategy) and PPS (Path Prefix Strategy) for ordering the branches, which are to be covered for testing. For improve test data they used elitism and memory with branch orderings. They compared each strategy with RAN and RNS and found best results with a mean number of generations and longer populations.

Andalib and Babamir [18], used PSO in discrete space for generating test data where there is no data dependency between program lines in a software. They proposed a method that produced minimum numbers of Test Case (TC) automatically with highest covering of codes in a program. In their method Mc Cabe theory was used to find the independent paths by reducing the number of paths in a program for selecting the best test case. Investing the motion of all the particles (birds/fish), the fitness function was taken for an optimal solution. They executed an integrated sorting program and with only one TC, they found 75% of the regions and 50% of independent paths are covered. The authors has compared their proposed algorithm with GA to covered 100% of independent paths and found more efficient result.

Dixit and Tomar [19], developed and implemented a hybrid algorithm GPSHA (Genetic Particle Swarm Hybrid Algorithm) combining the power of GA and PSO and they found a less number of generations and less number of test cases which covers around 100% of a program. Their results confirmed the effectiveness of the GPSHA over GA and PSO after performing in real world problems.

Sharma et al. [20], implemented GA in software testing to increase the efficiency and process time of testing. They generate test cases by using GA in Ruby, C++ and Matlab. It is found that the best fitness function is evaluated to a population of 50 and maximum generation 500. When the stopping condition is satisfied the iterative generation is stopped by providing an optimized and unique solution.

Yang et al. [21], *d*eveloped a new intelligent search based algorithm RGA (Regenerate Genetic Algorithm) to increase test coverage, search efficiency, restrain population aging and produce less number of test cases for coverage oriented software testing. They found RGA can give better optimized solution for large scale, highly complex problems and solve the population aging problem. After comparing with GA and random test method, authors found RGA is more efficient for required coverage criteria of test cases and achieving greater test coverage with fewer iterations and test cases.

Shahbazi and Miller [22], used a multi objective optimization in black box string test case generation for random testing and adaptive random testing. The authors performed their experiments by taking six different types of string distance functions such as Levenstein, Hamming, Cosine, Manhattan, Cartesian and Locality-Sensitive Hashing, to find effectiveness and run time of test cases. They introduce two objectives for effective string test cases such as the length distribution of the string test cases and the diversity control of the test cases within a test set. They used one diversity- based fitness function to generate optimized test sets to reveal faults more effectively and found superior test cases are produced by applying the objectives.

Zhenga et al. [23], developed a decomposition based multi-objective evolutionary algorithm (MOEA/D) for regression testing of programs from SIR repository. The experiments are evaluated in four approaches such as NSGA-II (non-dominated sorting genetic algorithm, MOEA/D (parameter c, used in normalization is fixed), MOEA/D (c is chosen from tuning) and classic greedy algorithm. The authors compared their work with Yoo and Harman [24] multi objective approaches where they used greedy algorithm and two versions of NSGA-II. They found among all the approaches MOEA/D with varying c is most effective and it produce the lowest HV(Hyper Volume) values with cheapest test suite. The two variants of MOEA/D have superior performance in comparison to NSGA-II and greedy algorithm.

After an extensive study of different evolutionary techniques used in software testing, we came to learn GA, PSO and BCO are used efficiently for generating test cases and solving many complex problems. Table 2 shows a brief summary of different evolutionary algorithm used in software testing and the results found in different related work has been already done.

Table 2. A brief summary of different evolutionary algorithm used in software testing and results found.

Authors	Problem discussed and solved	Algorithm used	Work done in particular area	Results
Mark Last et al. [11]	Generate test cases for mutation testing.	GA	Used a FLC (Fuzzy Logic Controller) for obtaining the probability of crossover	FAexGA is efficient as the rate of finding error is very fast and number of solution is distinct.
Hla et al. [12]	Prioritize test cases to increase effectiveness in regression testing for embedded real time software.	PSO	Focused on coverage based prioritization of test suite.	PSO algorithm can prioritized the test cases in very effectively and efficiently
McCaffrey [13]	Reducing test set in pair wise testing	BCO	Combinatorial NP hard and Pair wise Testing	Test cases are far better than the test sets generated by deterministic approach.
Nachiyappan et al. [14]	Decrease the cost of regression testing by reducing the test suite.	GA	APFD is used to determine the effectiveness of test cases.	The method is very highly adaptive as test case reduction is more effective with increase of fitness granularity.

(*continued*)

Table 2. (*continued*)

Authors	Problem discussed and solved	Algorithm used	Work done in particular area	Results
Kaur and Goyal [15]	Fault based test suit prioritization.	BCO	Used APFD (Average Percentage of Fault Detection) metrics and CPP compiler.	Maximum numbers of faults are covered in regression testing.
Ferrer et al. [16]	Test sequence generation with shortest valid path to cover transition and class.	GA & ABC	CIT (Combinatorial Interaction Testing), the extended classification tree method.	Generate test sequences with shortest valid path, which covers full transition and class in functional testing.
Ankur and Srivastav. [17]	Generate test data automatically for branch testing.	GA	Focused on branch ordering, memory and elitism.	Generate best results with a mean number of generations and longer populations.
Andalib and Babamir [18]	Generating minimum number of Test Case (TC) automatically with highest covering of codes in a program.	PSO	Used Mc Cabe theory to find the independent paths for selecting the best test case.	Covered 100% of independent paths and found more efficient result.
Dixit and Tomar [19]	Generation of Less and unique numbers of test cases.	GA & PSO	Combining the power of GA and PSO	GPSHA results a less number of generations and less number of test cases and covers around 100% of a program.
Sharma et al. [20]	Increase the efficiency and process time of testing.	GA	Generate test cases by using GA in Ruby, C ++ and Matlab.	Providing an optimized and unique solution for testing.

(*continued*)

Table 2. (*continued*)

Authors	Problem discussed and solved	Algorithm used	Work done in particular area	Results
Yang et al. [21]	Judging the population aging process.	GA	Used population regeneration strategy	RGA is more efficient by reducing the number of test cases and achieving greater test coverage with fewer iterations and test cases.
Shahbazi and Miller [22]	Generate effective set of black box string test cases through multi objective optimization.	GA & MOGA	Used several string distance functions to find effectiveness and run time of test cases.	Superior test cases are produced by using multi objective optimization technique.
Zhenga et al. [23]	To achieve full coverage for regression testing	MOEA & GA	Used MOEA/D with a normalization parameter c to solve multi objective optimization problem	MOEA/D have superior performance in comparison to NSGA-II and greedy algorithm.

3 An Introduction to Evolutionary Algorithm

Evolutionary algorithm based on biological behavior or evolution of population, which can be used to solve many complex and real life problems by producing high quality test data automatically [15, 23]. This algorithm is based on the principle of survival of the fittest and models some natural phenomena like genetic inheritance and Darwinian strife for survival, constitute an interesting category of modern heuristic search [9, 19]. Figure 1 shows the work flow of evolutionary technique.

3.1 Genetic Algorithm (GA)

GA has emerged as a practical, robust optimization technique and search method and it is inspired by the way nature evolves species using natural selection of the fittest individuals.

The algorithm was developed by John Holland in United States [14]. The solution to a specific problem can be solved by a population of chromosomes. A chromosome is

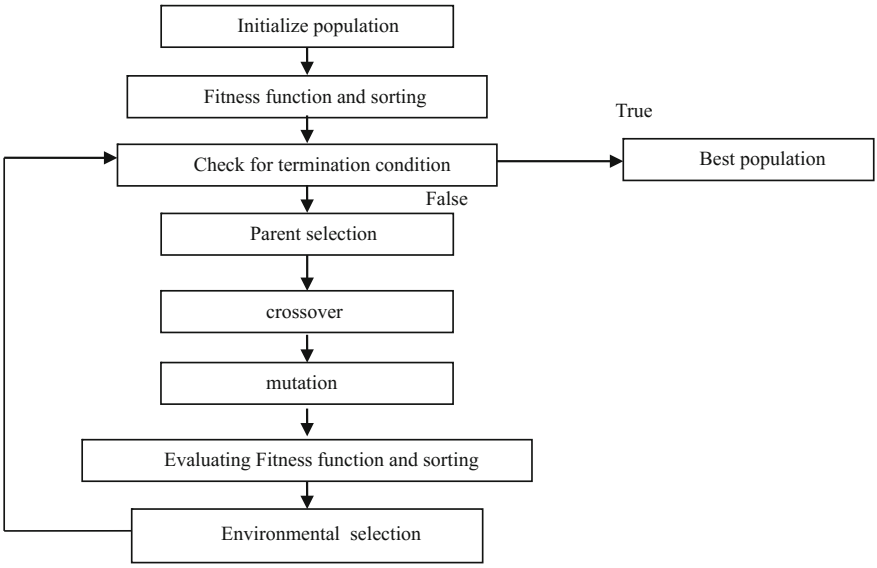

Fig. 1. Work flow of evolutionary technique

a string of binary digits and each digit is called a gene and population can be created randomly. It is a best way to solve optimization problems by searching for good genes and applying the different genetic operators like selection, crossover, mutation and Elitism [11, 17, 22].

Selection: A selection operation is performed to determine the individuals that meets the fitness function where fitness function is a specific function depending upon the criteria which returns a number indicating the acceptability of the program. This function is used in the selection process to determine the optimum point and the variants survive to the next iteration [8, 21]. Selection methods are of six different types such as roulette wheel, stochastic universal sampling, linear rank, exponential rank, binary tournament and truncation.

Crossover or Recombination: After selection, the crossover operation is applied to the selected chromosomes, which swaps genes or sequence of bits in the string between two individuals. For binary encoding different types of crossover operators are used like one point, two point, uniform and arithmetic. Cross over process is repeated with different parent individuals. Finally the mutation operator is applied to a randomly selected subset of the population [17, 20].

Mutation: It is used to maintain genetic diversity in the population by altering chromosomes to introduce new good traits. Basically six types of mutation operators are used in Genetic algorithm such as Bit string, flip bit, boundary, uniform, non uniform and Gaussian [17, 21].

Elitism: Elitism process involves copying a small proportion of the fittest candidates into the next generation, which are related to the best solution found [17].

The basic process of Genetic algorithms mainly involves creating an initial set of random solutions (population) and evaluating them [21, 23], by using the GA operators, in which the better solutions are identified (parents) and are then used to generate new solutions (children). These values can be used to replace with other population. This new population (generation), is then reevaluated and the process for generating new values continues until a final solution is found based on a specified condition of the fitness function [14]. Finally the function minimization is applied to the fitness function for test data generation.

3.2 Particle Swarm Optimization Algorithm (PSO)

PSO is a search based optimization technique that studies the social behavior of bird flocking or fish schooling. This algorithm mainly based on the movement and intelligence of swarms [18, 19]. The best solution can be found by a number of particles constituting a swarm, moving around in a particular search space of N-dimensional and adjusting their flying according to own and other's flying experience. Particles are always keeping track for personal best solution, denoted by *p-best* and the best value of any particle, denoted by *g-best*. Simultaneously the speed is adjusted dynamically of each particle depending on flying experiences. The velocity of each particle can be changed by considering the parameters like current position and velocity, distance between current position and its *p-best* as well as the distance between current position and its *g-best* [12, 19].

3.3 Bee Colony Optimization Algorithm (BCO)

Bee Colony Optimization (BCO) is a special type of Swarm Intelligence (SI), where the honey bees are the agents of the group. They communicate with each other by "Waggle Dance" principle to exchange information about the location for rich food source. In this system there is a well coordinated interaction between bees of a particular colony, organized team work and simultaneous task performance [13, 15].

In a bee colony different types of bees are present like a queen bee, many male drone bees and thousands of worker bees where the Queen is responsible to lay eggs for creating new colonies the male drones are responsible to mate with the Queen. At the time of downfall, male drones are discarded from the colony. The females of the hive are the worker bees. They are main responsible to build blocks of the hive as well as to comb, clean, maintain, guard the hive, search and collect rich food to feed the queen and drones. The worker bees are of two types such as forager bees and scout bees. The scout bees search food sources randomly and after finishing their distance limits they return back to the hive to give the information to foragers by "Waggle Dance" principle. Finally after observing the direction and information regarding location of rich food sources the foragers start flying to collect food [13, 24]. BCO algorithms are used to solve diverse domains problems, bench mark problems like routing problems, NP-hard problems and Travelling Salesman Problems [15].

4 Conclusion and Future Work

In this review paper we analyzed how different types of evolutionary techniques such as GA, PSO, ABCO and BCO have been efficiently used in software testing and have been applied extensively for automated test data generation. The results and performance of testing can be improved by these techniques. The evolutionary generation of test cases is proved to be very efficient and cost effective than manual testing. In future, we planned to combine the power of GA, PSO, ABCO and BCO in such a way that the new hybridized algorithm can produce a less number of test generations from which best test cases can be achieved for software testing. It is also planned to develop a new algorithm to generate test cases randomly and further optimize to find the best test cases.

References

1. Chauhan, N.: Software Testing: Principles and Practices. Oxford University Press, Oxford (2010)
2. Jogersen, P.C.: Software Testing: A Craftsman Approach, 3rd edn. CRC Presses, Boca Raton (2008)
3. Srivastava, P.R., Kim, T.H.: Application of genetic algorithm in software testing. Int. J. Softw. Eng. Appl. **3**(4), 87–96 (2009)
4. Berndt, D.J, Watkins, A.: High volume software testing using genetic algorithms. In: Proceedings of the 38th Annual Hawaii International Conference on System Sciences – Volume 09, vol. 9, pp. 318–326. IEEE Computer Society, Washington, DC (2005)
5. Wang, J., Changan, W., Shouda, J.: Test data generation algorithm of combinatorial testing based on differential evolution. In: Third International Conference on IEEE Instrumentation, Measurement, Computer, Communication and Control (IMCCC) (2013)
6. Vahid, G., Mäntylä, M.K.: When and what to automate in software testing? A Multi-Vocal Lit. Rev., Inf. Softw. Technol. **76**, 92–117 (2016)
7. Vudatha, C.P., Nalliboena, S., Jammalamadaka, S.K., Duvvuri, B.K.K., Reddy, L.: Automated generation of test cases from output domain of an embedded system using genetic algorithms. In: 3rd International Conference on Electronics Computer Technology (ICECT), vol. 5. IEEE (2011)
8. Sharma, C., Sabharwal, S., Sibal, R.: A survey on software testing techniques using genetic algorithm. arXiv preprint arXiv, pp. 1411–1154 (2014)
9. Wappler, S., Lammermann, F.: Using evolutionary algorithms for unit testing of object oriented software. In: GECCO, pp. 1925–1932. ACM (2005)
10. Goldberg, D.E: Genetic Algorithms: In Search, Optimization and Machine Learning. Addison Wesley, MA (1989)
11. Last, M., Eyal, S., Kandel, A.: Effective black-box testing with genetic algorithms. In: Ur, S., Bin, E., Wolfsthal, Y. (eds.) HVC 2005. LNCS, vol. 3875, pp. 134–148. Springer, Heidelberg (2006). doi:10.1007/11678779_10
12. Hla, K.H.S., Choi, Y., Park, J.S.: Applying particle swarm optimization to prioritizing test cases for embedded real time software retesting. In: IEEE 8th International Conference on Computer and Information Technology Workshops, CIT Workshops 2008, pp. 527–532. IEEE, July 2008

13. McCaffrey, J.D.: Generation of pair wise test sets using a simulated bee colony algorithm. In: IEEE International Conference on Information Reuse and Integration, IRI 2009. IEEE (2009)
14. Nachiyappan, S., Vimaladevi, A., Selva Lakshmi, C.B.: An evolutionary algorithm for regression test suite reduction. In: 2010 International Conference on Communication and Computational Intelligence (INCOCCI), pp. 503–508. IEEE, December 2010
15. Kaur, A., Goyal, S.: A survey on the applications of bee colony optimization techniques. Int. J. Comput. Sci. Eng. **3**(8), 30–37 (2011)
16. Ferrer, J., Kruse, P.M., Chicano, F., Enrique Alba, E.: Evolutionary algorithm for prioritized pairwise test data generation. In: Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation, pp. 1213–1220. ACM (2012)
17. Ankur, P., Srivastav, G.: Automated test data generation for branch testing using genetic algorithm: an improved approach using branch ordering, memory and elitism. J. Syst. Softw. **86**(5), 1191–1208 (2013)
18. Andalib, A., Babamir, S.M.: A new approach for test case generation by discrete particle swarm optimization algorithm. In: The 22nd Iranian Conference on Electrical Engineering (ICEE), May 20–22. Shahid Beheshti University (2014)
19. Dixit, S., Tomar, P.: Automated test data generation using computational intelligence, Reliability. In: 4th International Conference on Infocom Technologies and Optimization (ICRITO) (Trends and Future Directions). IEEE (2015)
20. Sharma, A., Rishon, P., Aggarwal, A.: Software testing using genetic algorithms. Int. J. Comput. Sci. Eng. Surv. (IJCSES) **7**(2), 21–33 (2016). doi:10.5121/ijcses
21. Yang, S., Man, T., Xu, J., Zeng, F., Li, K.: RGA: a lightweight and effective regeneration genetic algorithm for coverage-oriented software test data generation. Inf. Softw. Technol. **76**, 19–30 (2016)
22. Shahbazi, A., Miller, J.: Black-box string test case generation through a multi-objective optimization. IEEE Trans. Softw. Eng. **42**(4), 361–378 (2016)
23. Zheng, W., Hierons, R.M., Li, M., Liu, X., Vinciotti, V.: Multi-objective optimisation for regression testing. Inf. Sci. **334**, 1–16 (2016)
24. Yoo, S., Harman, M.: Regression testing minimization, selection and prioritization: a survey. Softw. Test. Verification Reliab. **22**(2), 67–120 (2012)