Review of Genetic Algorithm and Application in Software Testing

Patrick Kwaku Kudjo
Jiangsu University
China Zhenjiang
P.R.China, 212013

Elias Nii Noi Ocquaye
Jiangsu University
China Zhenjiang
P.R.China, 212013

Wolali Ametepe
Jiangsu University
China Zhenjiang
P.R.China, 212013

ABSTRACT
The recent technological advancement and complexity of software makes it very difficult to maintain and improve the quality of software. Software testing is a technique used in validating and verifying that a develop software meets it requirements and specification hence the need to properly generate suitable test case to test programs. Software testing consumers a lot of time, effort and money, therefore the focus of testing largely depends on test case generation, execution and evaluation. Automated testing is a technique used to maximize test coverage, detect more errors, increase test execution, decrease cost as well as improving the quality of software. This paper reviews the use of Genetic algorithm to automatically generate test case using the random method.

Keywords
Automated Testing; Genetic Algorithm; Evolutionary Algorithms; Random Testing; Test Case

1. INTRODUCTION
Software testing remains an integral part of the software development life cycle (SDLC); it is an important phase in the SDLC as it helps remove errors in different forms ranging from minor to major errors that can seriously affect the quality of product when delivered to the intended users. Previous research shows that, software testing phase takes a lot of effort, time and cost hence the need to find appropriate test case generation technique to effectively improve the testing process. Different test case generation techniques have been proposed in several research work with an objective of enhancing the testing process, these include random test data generators [1-3], path oriented test data generators[4-6], goal oriented test data technique[7], intelligent test data generator technique[8], other automatic test generation technique for java programs [9-15]. In this paper, we review the use of Genetic Algorithm to automatically generate test case using the simple random testing technique. Random testing is a simple and effective testing technique that allows a test case to be selected randomly from the input domain [16-18]. The uniform technique of random testing helps ensures that all inputs have the same probability of being selected from the input domain. The application of Genetic Algorithm (GA) in software testing is a young and active area of scientific research since much study has not been conducted in the field.

The paper review the application of Evolutionary algorithm (EA) specifically genetic algorithm based on test case generation using the simple random testing technique. An evolutionary algorithm (EA) is a stochastic search for an optimal solution to a given problem [19]. It uses a population of individuals; where each individual is classified as a chromosome. Chromosome represents the characteristics of individuals in the population; these characteristics are thus referred as gene[20, 21].

Genetic algorithm is one of the evolutionary algorithms that implements optimization strategies by simulating evolution of species through the process of natural selection, survival of the fittest and reproduction [22-24]. Genetic Algorithm is a subset of a branch of Evolutionary algorithms that include Evolutionary programing[25], evolutionary strategies[26, 27], differential evolution[28], cultural evolution and coevolution [29, 30]. This paper reviews the application of genetic algorithm in software testing specifically the generation of test case using the simple random testing technique. The structure of the paper is as follows, Section 2 will briefly give a background to genetic algorithm, application of genetic algorithm for the test case generation using the random technique is given in Section 3, Section 4 gives a detail analysis of Genetic Algorithm with other test case generation techniques, the related work is presented in Section 5 and Section 6 concludes and gives recommendations for future works.

2. GENETIC ALGORITHM
Genetic algorithm was proposed by John Holland and his fellow colleagues in Michigan University [31]. However the initial concept was first investigated by J.D. Bagley's in 1967 "The behavior of Adaptive systems which employ Genetic and Correlative Algorithms"[32]. Other independent study of Evolutionary algorithms include [33-38]. Genetic algorithm is a search based optimization technique [39-41] that evolve through a search space of candidate population to identify the best pool of population. To help understand the concept and its application in software testing, we will briefly explain some of the related concepts used in Genetic Algorithm.

A. Related Terms
i. Chromosome: are the strings or numerical values which represent the candidate solution to the search problem[42-44]

ii. Gene: the elements of the solution are known as the gene.

iii. Population: refers to the total number of candidate solutions to be used in the optimization problem.

iv. Allele: the possible values of a gene are known as allele.

v. Genotype refers to the set of genes contained in a genome.

vi. Phenotype: Are the behavioral traits or characteristics of an individual[45, 46].

B. *Pseudocode For Genetic Algorithm*

The general pseudocode is shown below

{

}

Initialize population;

Randomly generate *t* number of individuals;

Evaluate fitness of individuals;

While (fitness value! =termination criteria) do

{

Selection;

Crossover;

Mutation;

Calculate fitness function;

}

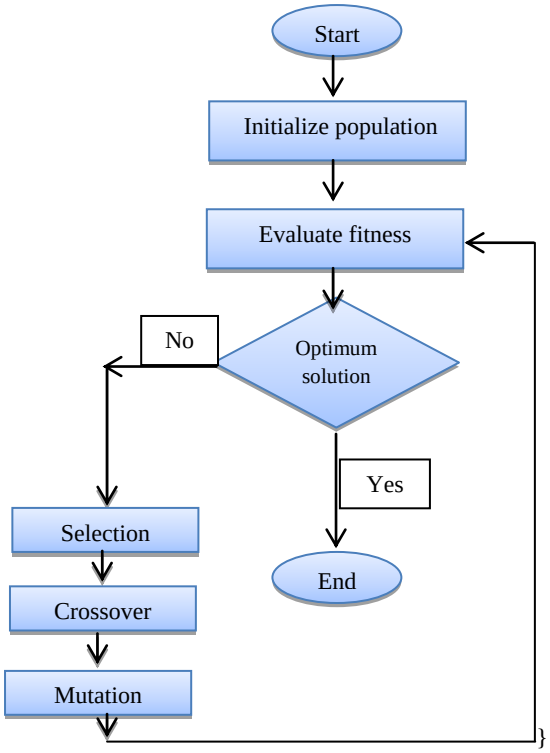

Fig 1: Flow chart of GA

2.1 Genetic Operators

Genetic algorithm uses three main operators which are selection, crossover and mutation.

2.1.1 Selection

The selection process determines which solutions are to be preserved and allowed to reproduce and which ones deserves to die out. The process normally involves identifying the good solutions in a population, making multiple copies of the good solutions, eliminating bad solutions from the population.[45, 47, 48]. The final stage is applying the fitness function to evaluate the solutions. A fitness function represents the quality of the solution[49, 50]. There are different types of selection methods used by GA, this include rank selection[51], proportionate selection[51, 52], roulette wheel selection [49], tournament selection [53], and steady selection

2.1.1.1 Crossover Operator

The crossover operator is used to create new solutions from the existing solutions by recombination of the information from both parents.[54]. It thus involves the exchange of genes randomly between the chromosomes of two parents to create new offspring [55, 56]. Crossover can either be one point or two points.

2.1.1.2 Mutation

The final genetic operator is mutation which introduces new features to a given population or problem to obtain and maintain diversity in the population. It thus involves randomly changing individuals to allow diversity among the population. The various types of mutation are Flip mutation, Uniform mutation inversion mutation, Scramble mutation, Swap mutation, interchanging mutation and Creep mutation [57-60]

3. TEST CASE GENERATION USING GENETIC ALGORITHM

Manual testing and automated testing are the two main testing techniques that have been employed over the past decades[61, 62], however automated testing appears to be the most efficient and effective method based on previous research[63], due partly to its efficiency and effectiveness in maximizing test coverage, detecting more errors, increasing test execution, decreasing cost and improving the quality of the product. The application of GA in test case generation usually starts with an initial population. Previous studies on the population size[64] indicate that, the size of the population used in any optimization problem could affect an optimization process, C. R. Reeves investigated the use GA with small population [65] and other studies that involve the use of large population are[66, 67] based on this, P. A. Diaz-Gomez and D. F. Hougen[68] proposed three types of metrics to help evaluate diversity of fixed length population of chromosomes. Based on the randomly generated test case from the population or input domain, a fitness function is used to evaluate the test cases to determine if it can be used for future generation of more test cases; the fitness function must be defined in relation to the testing goals. The test case is executed and feedback information is collected, all test cases with high fitness value are selected and added to the best test pool. Test cases with low fitness value are discarded and not used for future generation or can be recombine and mutated to produce new test cases, the newly created offspring are evaluated against the fitness value, this process continues until a stopping condition is achieved.

4. ANALYSIS OF GENETIC ALGORITHM WITH OTHER TEST CASE GENERATION TECHNIQUES

Based on previous studies conducted, genetic algorithm proves to be an effective technique for the generation of test case for detecting errors in complex systems[69] as compared to other test case generation techniques investigated by [10, 70] N. Kosindrdecha and J. Daengdej presented a number of test case generation process and techniques such as random testing technique, goal oriented techniques, specification-based techniques and sketch diagram based techniques, the findings shows a number of weakness related to these test data generation techniques. Some of the problems identified include the following:

i. Existing methods lacks the ability to identify and reserve the critical domain requirements in the test case generation process

ii. Inefficient automated test case generation techniques

iii. Failure to use most of the generated test cases

iv. Inefficient test case generation techniques with limited resources.

Although genetic algorithm has been considered an efficient and efficient method, D. J. Berndt and A. Watkins proposed two new strategies to help improve the performance of genetic algorithm:

i. Neural Network Based oracles: this method invovles replacing the real target sytem execution to avoid expensive execution costs for evaluating test cases.

ii. The second method invloves improving and enhancing the major genetic operators such as selection, crossover and mutation.

5. RELATED WORK

Genetic algorithm is applied in several fields of studies such as machine learning,[71, 72] business [40], Ecological[73] as well as medical data mining[74, 75]

Jarmo T. et al. [76] study program testing automation by using optimization via genetic algorithm. The findings of their investigation show that, genetic optimization performs better than simple random testing.

Rakesh Kumar et al. [77] showed how genetic algorithm can used to Automate test case generation. The findings shows that the use of genetic algorithm for full automation of test case generation enhance the overall quality of the software as compared with random testing.

Amit Kumar Sharma [26] also investigated how Genetic algorithm can be used to improve the quality and reliability of the software by generating optimized test cases, the results show that GA is a good technique in searching the input domain for the required test cases and as well improving the test case optimization process.

R.P. Pargas et.al [15] presented a GenerateData, an algorithm for automatic generation of test data using genetic algorithm directed by the control- dependence graph of the program, the approach adopted in the paper shows a potentially useful way of generating test data using the Genetic algorithm.

Kuvinder Singh et.al [78] generated test case using the genetic algorithm with anti-random population, experimental results of the works shows a significant average of 20% and maximum 40% improvement over random test case generation technique.

F. S. Babamir et al. [79] investigated the application of genetic algorithm based tester using different parameters to automate the structural oriented test data generation on the principle of internal program structure.

S. P. Tripathy and D. Kanhar proposed a heuristic algorithm with sampling techniques to optimize test suite. The genetic algorithm was used as an element to optimize the testsuite [80].

J. McCart et al. [81] used genetic algorithm to improve the testing technique process by reducing the execution time as well as improving the ability of the technique in detecting more error

N. K. Gupta and M. K. Rohil conducted a research by using the genetic algorithm to generate test cases for testing object oriented programs(OOP). The method uses a tree representation of statements in test cases. The approach adopted facilitates the automatic generation of object oriented programs using genetic algorithm[82].

S. Wappler and F. Lammermann [83] in their study proposed an evolutionary algorithm for the automatic generation of test cases for white- box testing of object-oriented programs. The experimental result of their approach outperforms random testing.

P. Tonella proposed a genetic algorithm to automatically generate test case to test classes in Java programs. The approach employed effectively achieves good coverage. [84]

A. Rauf et al. [85] proposed a graphical User Interface (GUI) testing and coverage analysis technique using genetic algorithm to help generate the best possible test parameter based on a predefined testing goals. They used some of the modern tools and techniques of automating GUI testing such as control- flow graph, even-flow graph and event-flow model etc. to help reduce the challenges associated with testing of such systems.

B. F. Jones [86] also investigated structural testing using genetic algorithm.

6. CONCLUSION

The work review various application of Genetic Algorithm in software testing specifically automated test generation. The ultimate aim of testing is to produce quality programs that meet the needs of customer's hence the need to select appropriate test generation techniques to help improve the testing process. Based on the study conducted we can conclude that, the application of Genetic Algorithm as an optimization technique to automatically generate test case helps detect more errors in program as it uses the best possible test case. The findings further shows that, Genetic algorithm performs better than other automated test generation techniques such as random testing and also increases the number of test cases thereby improving efficiency. Future study will be focus on investigating the application of GA and Particle Swarm Optimization (PSO) in test case automation. Another area of study that can be investigated is the use of Genetic algorithm in regression testing as well as test case prioritization. Finally a more robust way of improving upon the fitness function can be investigated.

7. ACKNOWLEDGMENTS

Our sincere thanks go to all the Professors and PhD students in the Computer Science and Communication Engineering Department of Jiangsu University China.

8. REFERENCES

[1] H. D. Mills, M. Dyer, and R. C. Linger, "Cleanroom software engineering," *IEEE Software*, vol. 4, p. 19, 1987.

[2] J. Voas, L. Morell, and K. Miller, "Predicting where faults can hide from testing," *IEEE Software*, vol. 8, pp. 41-48, 1991.

[3] P. Thevenod-Fosse and H. Waeselynck, "STATEMATE applied to statistical software testing," in *ACM SIGSOFT Software Engineering Notes*, 1993, pp. 99-109.

[4] L. A. Clarke, "A system to generate test data and symbolically execute programs," *IEEE Transactions on software engineering*, pp. 215-222, 1976.

[5] R. S. Boyer, B. Elspas, and K. N. Levitt, "SELECT—a formal system for testing and debugging programs by symbolic execution," *ACM SigPlan Notices*, vol. 10, pp. 234-245, 1975.

[6] R. DeMilli and A. J. Offutt, "Constraint-based automatic test data generation," *IEEE Transactions on Software Engineering*, vol. 17, pp. 900-910, 1991.

[7] B. Korel, "Automated software test data generation," *IEEE Transactions on software engineering*, vol. 16, pp. 870-879, 1990.

[8] I. Hermadi and M. A. Ahmed, "Genetic algorithm based test data generator," in *Evolutionary Computation, 2003. CEC'03. The 2003 Congress on*, 2003, pp. 85-91.

[9] M. Prasanna, S. Sivanandam, R. Venkatesan, and R. Sundarrajan, "A survey on automatic test case generation," *Academic Open Internet Journal*, vol. 15, 2005.

[10] N. Kosindrdecha and J. Daengdej, "A test case generation process and technique," *J. Software Eng*, vol. 4, pp. 265-287, 2010.

[11] N. Kaushik, K. Choudhary, and N. S. Yadav, "Year of Publication: 2016."

[12] N. Kosindrdecha and J. Daengdej, "A TEST CASE GENERATION TECHNIQUE AND PROCESS," in *International Workshop on Design, Evaluation and Refinement of Intelligent Systems (DERIS2010)*, 2010, p. 59.

[13] M. G. Xu, Y. M. Mu, Z. H. Zhang, and A. Liu, "Research on Automatic Test Case Generation Framework for Java," in *Applied Mechanics and Materials*, 2014, pp. 1488-1496.

[14] H.-H. Sthamer, "The automatic generation of software test data using genetic algorithms," University of Glamorgan, 1995.

[15] R. P. Pargas, M. J. Harrold, and R. R. Peck, "Test-data generation using genetic algorithms," *Software Testing Verification and Reliability*, vol. 9, pp. 263-282, 1999.

[16] J. H. Andrews, A. Groce, M. Weston, and R.-G. Xu, "Random test run length and effectiveness," in *Proceedings of the 2008 23rd IEEE/ACM International Conference on Automated Software Engineering*, 2008, pp. 19-28.

[17] G. Myers, "The Art of Software Testing. revised and updated by T. Badgett and TM Thomas with C. Sandler," ed: John Wiley & Sons, Inc, 2004.

[18] R. Hamlet, "Random testing," *Encyclopedia of software Engineering*, 1994.

[19] Z. W. Geem, J. H. Kim, and G. Loganathan, "A new heuristic optimization algorithm: harmony search," *Simulation*, vol. 76, pp. 60-68, 2001.

[20] D. Beasley, D. R. Bull, and R. R. Martin, "An overview of genetic algorithms: Part 1, fundamentals," *University computing*, vol. 15, pp. 56-69, 1993.

[21] H. Bati, L. Giakoumakis, S. Herbert, and A. Surna, "A genetic approach for random testing of database systems," in *Proceedings of the 33rd international conference on Very large data bases*, 2007, pp. 1243-1251.

[22] S. Suman, "A genetic algorithm for regression test sequence optimization," *Int. J. Adv. Res. Comput. Commun. Eng*, vol. 1, 2012.

[23] C. Sharma, S. Sabharwal, and R. Sibal, "A survey on software testing techniques using genetic algorithm," *arXiv preprint arXiv:1411.1154*, 2014.

[24] E. D. Goodman, "Introduction to genetic algorithms," in *Proceedings of the Companion Publication of the 2014 Annual Conference on Genetic and Evolutionary Computation*, 2014, pp. 205-226.

[25] L. J. Fogel, A. J. Owens, and M. J. Walsh, "Artificial intelligence through simulated evolution," 1966.

[26] H.-G. Beyer and H.-P. Schwefel, "Evolution strategies–A comprehensive introduction," *Natural computing*, vol. 1, pp. 3-52, 2002.

[27] I. Rechenberg, "Evolution Strategy: Optimization of Technical systems by means of biological evolution," *Fromman-Holzboog, Stuttgart*, vol. 104, 1973.

[28] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE transactions on Evolutionary Computation*, vol. 13, pp. 398-417, 2009.

[29] K. E. Kinnear Jr, "A perspective on the work in this book," *Advances in Genetic Programming*, pp. 3-19, 1994.

[30] S. Shennan, "Genes, memes, and human history: Darwinian archaeology and cultural evolution," 2002.

[31] J. H. Holland, "Genetic algorithms," *Scientific american*, vol. 267, pp. 66-72, 1992.

[32] J. D. Bagley, "The behavior of adaptive systems which employ genetic and correlation algorithms," 1967.

[33] G. J. Friedman, "Digital simulation of an evolutionary process," *General Systems Yearbook*, vol. 4, 1959.

[34] D. B. Fogel, "An introduction to simulated evolutionary optimization," *IEEE transactions on neural networks*, vol. 5, pp. 3-14, 1994.

[35] K. De Jong, "Genetic algorithm based learning," 1990.

[36] K. A. De Jong, "Genetic algorithms are NOT function optimizers," *Foundations of genetic algorithms*, vol. 2, pp. 5-17, 1993.

[37] H. J. Bremermann, *The evolution of intelligence: The nervous system as a model of its environment*: University of Washington, Department of Mathematics, 1958.

[38] E. Falkenauer, *Genetic algorithms and grouping problems*: John Wiley & Sons, Inc., 1998.

[39] D. E. Goldberg, "Genetic algorithms in search, optimization and machine learning ‘addison-wesley, 1989," *Reading, MA*, 1989.

[40] L. Davis, "Handbook of genetic algorithms," 1991.

[41] J. A. Joines and C. R. Houck, "On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GA's," in *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*, 1994, pp. 579-584.

[42] M. Mitchell, "Genetic algorithms: An overview," *Complexity*, vol. 1, pp. 31-39, 1995.

[43] J. F. Gonçalves, J. J. de Magalhães Mendes, and M. c. G. Resende, "A hybrid genetic algorithm for the job shop scheduling problem," *European journal of operational research*, vol. 167, pp. 77-95, 2005.

[44] K. Sastry, D. E. Goldberg, and G. Kendall, "Genetic algorithms," in *Search methodologies*, ed: Springer, 2014, pp. 93-117.

[45] M. Mitchell, *An introduction to genetic algorithms*: MIT press, 1998.

[46] A. P. Engelbrecht, *Fundamentals of computational swarm intelligence*: John Wiley & Sons, 2006.

[47] S. Sivanandam and S. Deepa, *Introduction to genetic algorithms*: Springer Science & Business Media, 2007.

[48] J. Stender, "Introduction to genetic algorithms," in *Applications of Genetic Algorithms, IEE Colloquium on*, 1994, pp. 1/1-1/4.

[49] S. A. Kazarlis, A. Bakirtzis, and V. Petridis, "A genetic algorithm solution to the unit commitment problem," *IEEE transactions on power systems*, vol. 11, pp. 83-92, 1996.

[50] A. Baresel, H. Sthamer, and M. Schmidt, "Fitness function design to improve evolutionary structural testing," in *Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation*, 2002, pp. 1329-1336.

[51] D. E. Goldberg and K. Deb, "A comparative analysis of selection schemes used in genetic algorithms," *Foundations of genetic algorithms*, vol. 1, pp. 69-93, 1991.

[52] M. Srinivas and L. M. Patnaik, "Genetic algorithms: A survey," *Computer*, vol. 27, pp. 17-26, 1994.

[53] B. L. Miller and D. E. Goldberg, "Genetic algorithms, tournament selection, and the effects of noise," *Complex systems*, vol. 9, pp. 193-212, 1995.

[54] M. Srinivas and L. M. Patnaik, "Adaptive probabilities of crossover and mutation in genetic algorithms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 24, pp. 656-667, 1994.

[55] U. Bodenhofer, "Genetic algorithms: theory and applications," ed: Lecture notes, Fuzzy Logic Laboratorium Linz-Hagenberg, Winter, 2003.

[56] C. Reeves, "Genetic algorithms," in *Handbook of metaheuristics*, ed: Springer, 2003, pp. 55-82.

[57] N. Soni and T. Kumar, "Study of various mutation operators in genetic algorithms," *IJCSIT) International Journal of Computer Science and Information Technologies*, vol. 5, pp. 4519-4521, 2014.

[58] L. Davis, "Applying adaptive algorithms to epistatic domains," in *IJCAI*, 1985, pp. 162-164.

[59] D. E. Goldberg, *Genetic algorithms*: Pearson Education India, 2006.

[60] D. Beasley, D. R. Bull, and R. R. Martin, "An overview of genetic algorithms: Part 2, research topics," *University computing*, vol. 15, pp. 170-181, 1993.

[61] J. Itkonen, M. V. Mantyla, and C. Lassenius, "How do testers do it? An exploratory study on manual testing practices," in *Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement*, 2009, pp. 494-497.

[62] C. M. Miller, "Automated testing system," ed: Google Patents, 1995.

[63] C. C. Michael, G. E. McGraw, M. A. Schatz, and C. C. Walton, "Genetic algorithms for dynamic test data generation," in *Automated Software Engineering, 1997. Proceedings., 12th IEEE International Conference*, 1997, pp. 307-308.

[64] D. E. Goldberg, *Optimal initial population size for binary-coded genetic algorithms*: Clearinghouse for Genetic Algorithms, Department of Engineering Mechanics, University of Alabama, 1985.

[65] C. R. Reeves, "Using Genetic Algorithms with Small Populations," in *ICGA*, 1993, p. 92.

[66] C. W. Ahn and R. S. Ramakrishna, "A genetic algorithm for shortest path routing problem and the sizing of populations," *IEEE transactions on evolutionary computation*, vol. 6, pp. 566-579, 2002.

[67] J. Horn, N. Nafpliotis, and D. E. Goldberg, "A niched Pareto genetic algorithm for multiobjective optimization," in *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*, 1994, pp. 82-87.

[68] P. A. Diaz-Gomez and D. F. Hougen, "Initial Population for Genetic Algorithms: A Metric Approach," in *GEM*, 2007, pp. 43-49.

[69] D. J. Berndt and A. Watkins, "Investigating the performance of genetic algorithm-based software test case generation," in *High Assurance Systems Engineering, 2004. Proceedings. Eighth IEEE International Symposium on*, 2004, pp. 261-262.

[70] A. Sharma, R. Patani, and A. Aggarwal, "SOFTWARE TESTING USING GENETIC ALGORITHMS."

[71] D. E. Goldberg and J. H. Holland, "Genetic algorithms and machine learning," *Machine learning*, vol. 3, pp. 95-99, 1988.

[72] K. A. De Jong, W. M. Spears, and D. F. Gordon, "Using genetic algorithms for concept learning," in *Genetic Algorithms for Machine Learning*, ed: Springer, 1993, pp. 5-32.

[73] A. Peterson, J. Soberón, and V. Sánchez-Cordero, "Conservatism of ecological niches in evolutionary time," *Science*, vol. 285, pp. 1265-1267, 1999.

[74] M. Brameier and W. Banzhaf, "A comparison of linear genetic programming and neural networks in medical data mining," *IEEE Transactions on Evolutionary Computation*, vol. 5, pp. 17-26, 2001.

[75] E. A. Sconce, T. I. Khan, H. A. Wynne, P. Avery, L. Monkhouse, B. P. King, *et al.*, "The impact of CYP2C9 and VKORC1 genetic polymorphism and patient characteristics upon warfarin dose requirements: proposal for a new dosing regimen," *Blood*, vol. 106, pp. 2329-2333, 2005.

[76] J. T. Alander and T. Mantere, "Automatic software testing by genetic algorithm optimization, a case study," in *Proceedings of the 1st International Workshop on Soft Computing Applied to Software Engineering*, 1999, pp. 1-9.

[77] R. Kumar, S. Singh, and G. Gopal, "Automatic Test Case Generation Using Genetic Algorithm," *International Journal of Scientific & Engineering Research (IJSER)*, vol. 4, pp. 1135-1141, 2013.

[78] K. Singh and R. Kumar, "Optimization of functional testing using Genetic algorithms," *International Journal of Innovation, Management and Technology*, vol. 1, p. 43, 2010.

[79] F. S. Babamir, A. Hatamizadeh, S. M. Babamir, M. Dabbaghian, and A. Norouzi, "Application of genetic algorithm in automatic software testing," in *International Conference on Networked Digital Technologies*, 2010, pp. 545-552.

[80] S. P. Tripathy and D. Kanhar, "Optimization of Software Testing for Discrete Testsuite using Genetic Algorithm and Sampling Technique," *International Journal of Computer Applications*, vol. 63, 2013.

[81] J. McCart, D. Berndt, and A. Watkins, "Using genetic algorithms for software testing: Performance improvement techniques," *AMCIS 2007 Proceedings*, p. 222, 2007.

[82] N. K. Gupta and M. K. Rohil, "Using genetic algorithm for unit testing of object oriented software," in *2008 First International Conference on Emerging Trends in Engineering and Technology*, 2008, pp. 308-313.

[83] S. Wappler and F. Lammermann, "Using evolutionary algorithms for the unit testing of object-oriented software," in *Proceedings of the 7th annual conference on Genetic and evolutionary computation*, 2005, pp. 1053-1060.

[84] P. Tonella, "Evolutionary testing of classes," in *ACM SIGSOFT Software Engineering Notes*, 2004, pp. 119-128.

[85] A. Rauf, S. Anwar, M. A. Jaffer, and A. A. Shahid, "Automated GUI test coverage analysis using GA," in *Information Technology: New Generations (ITNG), 2010 Seventh International Conference on*, 2010, pp. 1057-1062.

[86] B. F. Jones, H.-H. Sthamer, and D. E. Eyres, "Automatic structural testing using genetic algorithms," *Software Engineering Journal*, vol. 11, pp. 299-306, 1996.