Automatic Software Structural Testing by Using Evolutionary Algorithms for Test Data Generations

Maha Alzabidi† , Ajay Kumar††, and A.D. Shaligram†††

†Department of Computer Science, University of Pune, Pune-411007, India
††JSPM's Jaywant Institute of Computer Applications, Pune-411033, India
†††Department of Electronic Sciences, University of Pune, Pune-411007, India

Summary
Software testing is an important activity of the software development process. It is a critical element of software quality assurance. Structural-oriented test methods which define test cases on the basis of internal program structure are widely used. Evolutionary testing is a promising approach for automation of structural test case design, which search test data that fulfill given structural test criteria by manner of evolutionary computation.
In this article we investigate the performance of proposed GA with different parameters combinations used to automate the test data generation for path coverage. The investigation involves crossover strategies and methods of selecting of parents for reproduction and mutation rates. The results of the study showed that double crossover was more successful in path coverage. The study results Also that, selecting parent for reproduction according to their fitness is more efficient than random selection. And that mutation rate is better adjusted with program at hand.
Also, we studied the generation to generation progress in the proposed GA while searching for good test data. The work is compared with random testing. And we concluded that the proposed GA improves the search from one generation to the next, and performs better than random testing, where the search was absolutely random and does not show improvement through the generations. Another observation is that random testing generates less successful test cases than the proposed GA.
Key words:
software testing - unit testing – path testing - genetic algorithms - test data generation.

1. Introduction

Software testing is a laborious and time-consuming work [1] [18]. It spends almost 50% of the software system development resources. To increase the effectiveness and efficiency of the test and thus to reduce the overall development cost for software system, we require a test that is systematic and automatable. The automation of test case generation is the most important aspect of automatic testing. No powerful test data generation tools are commercially available today.

Manuscript received April 5, 2009
Manuscript revised April 20, 2009

For years, many researchers have proposed different methods for generating test data automatically [3][4][5] [6][7]. This article studies the use of GA in structural testing, where test cases are selected so that the structural components are covered.
The study is based on path coverage criterion for testing.
In section 2 we explain the objectives of the work then a short overview of evolutionary testing and path testing is presented in section 4 and 4 respectively. Our proposed GA is described in section 5. The results of the conducted experiments are presented section 6. Conclusion and remarks are given in section 7.

2. Objective

Past experiment with GA indicates that GA needs to be tuned for the problem of test data generation. In this article we investigate the performance of different GA parameters. The investigation of the proposed GA includes the following: (1) which crossover strategy is most efficient. (2) which parent selection method is most efficient. (3) which mutation rate gives best result.

3. Evolutionary testing

Evolutionary testing is characterized by the use of meta-heuristic search for test case generation. The considered test aim is transformed into an optimization problem [6] [8] where the input domain of the test object forms the search space for test data that fulfills the respected search aim. An example of evolutionary algorithms is Genetic Algorithms (GA). GA operates on a string of digits called chromosomes [9], each digit that makes up the chromosome is called gene, and a collection of such chromosomes makes up a population. Each has a fitness value associated with it, and this fitness value determines the probability of survival of an individual to the next generation. After the next generation is created a percentage of the chromosomes are crossed and small

percentages are mutated. A block diagram of GA is shown in Fig.1.

GA with selection, crossover, and mutation is an adoptive search technique that has been applied in many areas [9]. Adoptive search techniques are not guaranteed to find the optimal solution, however they often find a very good solution in limit of time [10]. GA is used to generate test data because their robustness and suitability for solutions of different test tasks has already been proven in previous work, e.g. [3], [7], [11], and [12].

Genetic algorithms guarantee high probability of improving the quality of individuals over several generations according to the Schema Theorem [10].

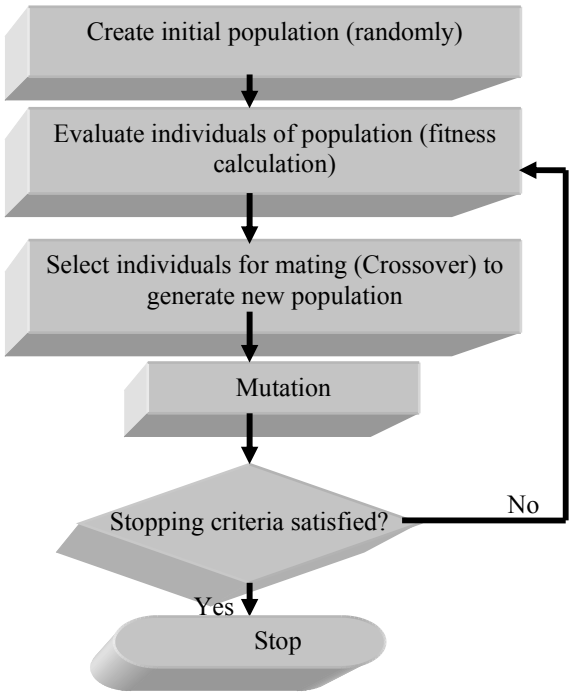

Fig .1. Block diagram of Genetic Algorithms

4. Path testing

Path coverage is an effective software structural testing criteria. If testing can be designed to force execution of all paths, every statement in the program will have been guaranteed to be executed at least one time and every condition will have been executed in its true and false sides [5] which make it the utmost coverage. Thus an effective software structural testing should have path coverage as its objective.

Genetic algorithms could be applied to path testing if the target paths are clearly defined and an appropriate fitness function related to this goal is built.

Test case generation for path testing consists of four basic steps:

1) Control Flow Graph construction: A CFG is a representation of a program where contiguous regions of code without branches, known as basic blocks, are represented as nodes in a graph and edges between nodes indicate the possible flow of the program. A cycle in CFG may imply that there is a loop in the code. In this step, the source program is transferred to a graph that represents the control flow of the program. Each branch of the graph is denoted by a label and different branches correspond with different labels.

2) Target path selection: In path testing, paths are extracted from the CFG, and some paths might be very meaningful and need to be selected as target path for testing (e.g. the path for Equilateral triangle).

3) Test case generation and execution: in this step the algorithm automatically creates new test cases to execute new path and leads the control flow to the target path. Finally, a suitable test case that executes the target paths could be generated.

4) Test result evaluation: this step is to execute the selected path and to determine the test criteria is satisfied

5. The Proposed Algorithm

The aim of this work is improving fitness function calculation, which is the more costly component of genetic algorithm, to get performance improvements. Our approach is to develop GA algorithm to generate test data for path coverage. The fitness function proposed to evaluate each test data is a modification to the Hamming Distance, which quantify the distance between two given paths. The fitness function in this work will be named Shifted-Modified-SIMILARITY (SMS). Given a target path and a current one, the similarity is calculated from cascaded edges for each of the paths being compared using hamming distance or symmetric differences (the symmetric difference between set A and set B is ( )). Similarities are then normalized and summed associated with a weighting factor which is usually found by experience. To obtain the total similarity this value is used as the objective function used to evaluate individuals in the population.

For path testing, two different paths may contain the same branches but in different sequences [13]. To increase path coverage within limited time and computational efforts, we proposed the (SMS). Where we shift paths being compared to right and left for calculating that distance (detailed in [14]). To obtain the objectives of the work we experimented in the following areas:

1) GA Parameter Combinations: two crossover strategies, namely single-point and double-point crossover (illustrated in Fig.2.), were studied. Also, selection of parents for reproduction at random an according to fitness are two methods investigated, and results are discussed in the next section. In addition, probability of mutation (Pm) is another GA parameter which was varied and applied to find the suitable Pm rate that gives best results. The results discuss different Pm rates.

2) Monitoring GA progress: in this part we study the speed of convergence of the population generated from generation to generation. This measures the efficiency of the fitness function and the difficulty of the target/goal path.

Single-point crossover at k=9

Parents	**1100101010**	**0101010111**	**0000101100**
	0011100001	*1111100000*	*1010101010*
Offspring	**1100101001**	*1111100000*	*1010101010*
	0011100010	**0101010111**	**0000101100**

Double-point crossover at k=5, m=13

Parents	**1100101010**	**0101010111**	**0000101100**
	0011100001	*1111100000*	*1010101010*
Offspring	**1100***100001*	*11***01010111**	**0000101100**
	*0011***101010**	**01***111100000*	*1010101010*

Fig.2. Single-point and double- point crossover.

6. Discussion of Results

To investigate the performance of GA parameter combinations with the proposed algorithm, we experimented with the triangle classifier program (Tri-Class). The program is a benchmark for many researches in software testing [15] [16] [17][18]. It classifies a triangle as valid, scalene, isosceles or equilateral given the length of the triangle sides. In the first step, the selection of parents for reproduction was either random or according to fitness (roulette wheel) method. Fig.3 list the average of test data generated to cover the goal paths in the program over 10 runs. The population was 500 individuals for 50 generations/iterations with 1.0 probability of single-point crossover and mutation rate Pm= 0.05. As shown, selection of parent according to their fitness value was far more effective than random selection.

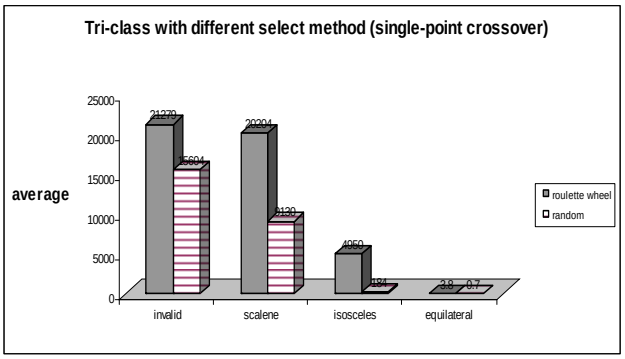

Fig.3. Path coverage for Tri-Class with different select method of parents

Fig.4 shows the same comparison using same parameters in the above experiment except that double-point crossover was used to generate offspring. As can be seen in the figure, the selection of parents according to fitness value was more successful in generating test data to cover all goal paths in the program under test than random method. In general, when the generation of the next population based on roulette wheel, the fitness function determines the probability of selection allowing those with high fitness more chance of offspring in the next generation in comparison of their less fit companions.

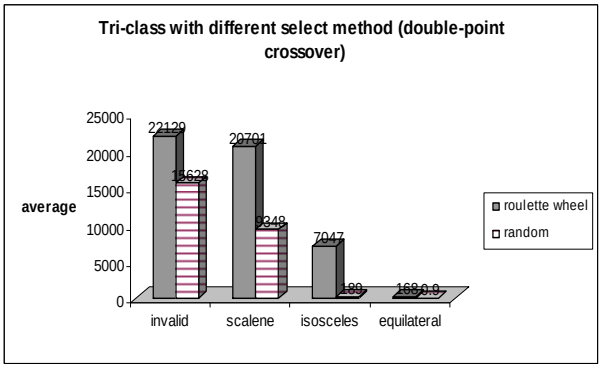

Fig.4. Path coverage for Tri-Class with double-point crossover.

In another experiment, the mutation rate for the proposed algorithm was investigated. Fig.5 shows the experiment results with two mutation rates. The first is 0.05 which is the reciprocal of the chromosome length $(\frac{1}{24})$ and the second was a smaller value of (0.005) which approximately is the reciprocal of generation population size. As can be seen in Fig.5, the lower rate of 0.005 for Pm was better than 0.05 which was less effective.

The best mutation rate for the Tri-Class program turns to be 0.005. Which means that on the average, one mutation

every 200th gene. If mutation rate was high, the change in the chromosome is too big; the building block of the chromosome will be lost or badly affected. It will be similar or close to random generation of data, and that is why results are worsen when experimented with Pm rates: 1.0 and 0.1. In other experiments, when Pm is very small (like 0.001, 0.0001) the results deteriorate because there is no enough disruption in the population.

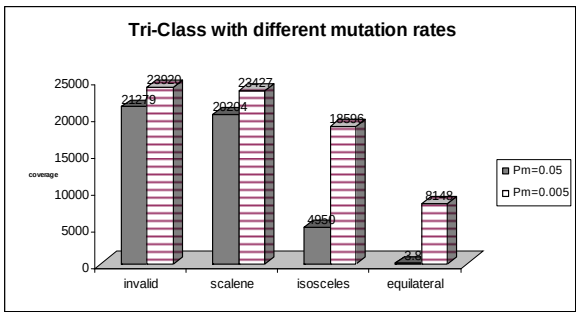

Fig.5. Path coverage in Tri-Class using different mutation rates

Another interesting result is shown in Fig.6. The figure shows comparison of GA performance in both single-point crossover and double-point crossover, the other parameters was same as before, i.e. 500 individuals for 50 generations and Pm=0.005. As can be seen in the figure, the average number of data generated to cover paths for the four types of triangle in double-point crossover was larger than the single-point crossover. This is because in double-point crossover there are good chances to double exchange the good genes between two fit parent to generate a yet better offspring. In the case of Tri-Class program, Comparison between the two crossover operators favors double-point crossover. Single-point-crossover may only change one input variable at a time, but double-point crossover may change two input variables (see Fig.2. above), which increase the effectiveness of the search. In general double-point crossover explores more of the domain than single-point crossover. This gives the indication that the strength of GA techniques exists in crossover operation.

In this section we show results of studying generation to generation progress in the GA search i.e. how does case-generation progresses from one generation/iteration to the next. The proposed GA algorithm using SMS as objective fitness function was applied to the Max-Min and Tri-Class programs. Tri-Class was explained earlier. Max-Min (the maximum minimum) program finds the maximum and minimum values, within an array of numbers. The program has two sequential selection statements inside a loop in which all the conditions are simple. And to allow more confidence in the proposed algorithm, the obtained result is compared with results obtained from random testing for the same two programs with similar parameters.

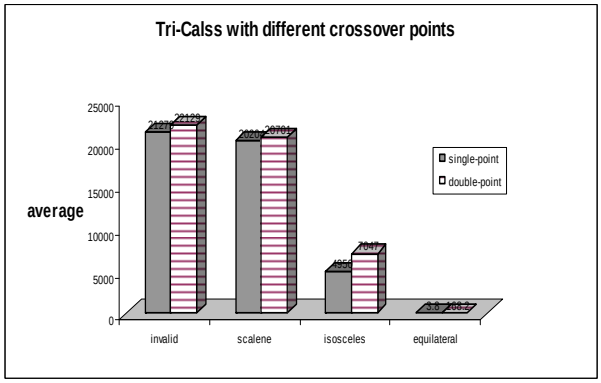

Fig.6. Path coverage in Tri-Class using different crossover points

The GA parameters used for MAX-Min program was as follows: pop size of 500, max-gen of 10 Pc=1.0 (single-point) Pm=0.05. The parent selection was according to fitness. For simplification, the results of selected 3 paths out of 13 paths in Max-Min program, over 10 generation, is shown in Fig.7-a. As shown in the figure, the selected goal paths were covered with good amount of data, since they are equal in difficulty of coverage and the progress is visible from 1st generation. SMS performs better and better from one generation to the next. In Fig.7-b., the progress in random testing is rather random. No improvement in the search for test data is seen in the selected 3 paths. Another observation from the results (shown in Fig.7-a. and Fig.7-b.) is that the number of good test data generated using SMS is about double the data generated using random testing.

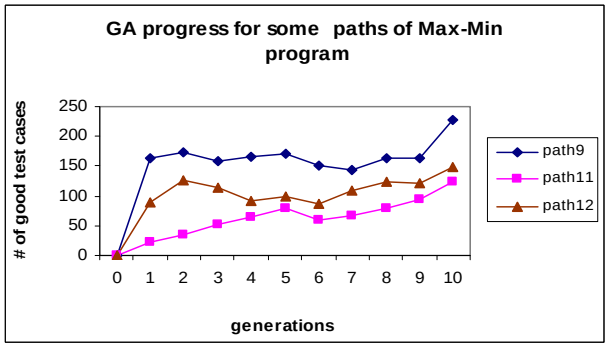

Fig.7-a. GA performance progress for Max-Min

Same results were obtained with Tri-Class program using population of 500 individuals over 50 generations (see Fig.8-a. and Fig.8-b).

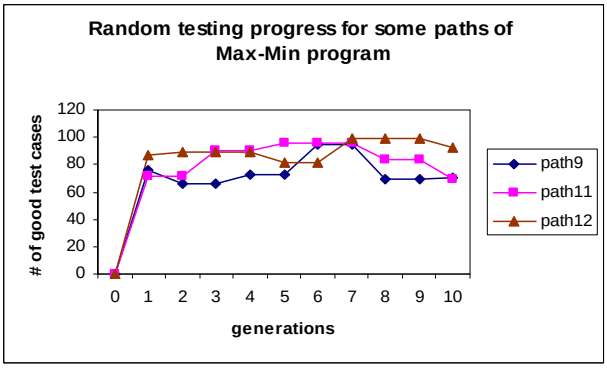

Fig.7-b. Random testing performance progress for Max-Min

Fig.8-a. shows that invalid and scalene triangle types were easily covered with large amount of good test data to cover them in the 1st to the 3rd generation (and then the average is approximately 400 test cases in each generation) while it took SMS about 10 generation to improve and give an adequate number of good test data to cover Isosceles triangle path (approximately 150 test cases in each generation). The most difficult path to be covered in the program is the equilateral triangle. The SMS shows improvement after about 30 generations/iterations since it is very difficult to generate test cases with 3 identical numbers to cover the path. Fig.8-b. shows random test progress for Tri-Class program with same number of individuals and generations. The progress is not clear in invalid and scalene triangles paths though they are comparatively easy to cover, while the progress is poor and there is almost no progress in generating test data for the difficult paths of isosceles and equilateral triangles. And generally, the average number of good test data generated is less than (approximately half) the average generated using SMS-based GA, in all the selected goal paths.

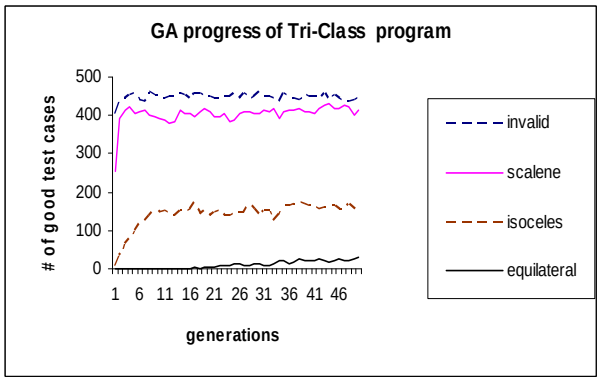

Fig.8-a. GA performance progress for Tri-Class

We conclude that the random testing does not improve from one generation to the next. the search is absolutely random in each iteration and good test data generated are less than SMS-based GA. This indicates that calculated SMS fitness value provides the necessary feed back for the GA to generate better test data in the next generations.

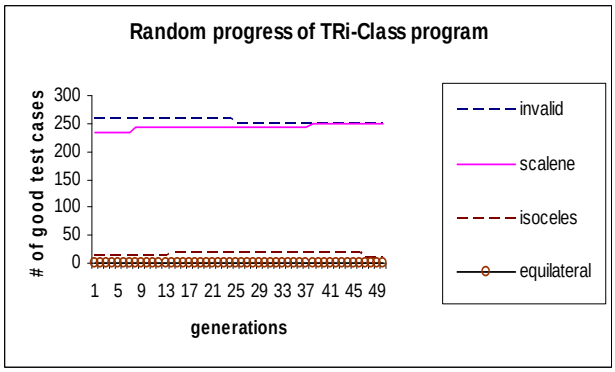

Fig.8-b. Random testing performance progress for Tri-Class

7. Conclusion

This article investigates the performance of a proposed GA for path testing. We have studied the performance of different GA parameters (crossover strategies, mutation rates and parent selection methods). The experiments with Tri-Class program showed that double crossover is more effective than single cross over strategy. And that selection parent for reproduction according to their fitness gave better results than random selection of parents. A very small value for mutation rate (0.005) gave the best results. In fact, larger rates for Pm were tested but were found disruptive while much lower rates was less effective which indicates that mutation rate (Pm) is better adjusted with program at hand.

For studying the GA progress from generation to generation, the experiment with Tri-Class and Max-Min programs showed that SMS-based GA performs better from one generation to the next to cover selected paths even with difficult paths like Equilateral triangle in Tri-Class program. The comparison with random testing showed that when generation of the next population is based on selecting those individuals with high fitness, this gives the chance to generate a better offspring than of random testing which had a poor or no progress in the generation of test data and generated much less number of good test data.

The next step is to experiment the proposed algorithm using more complex programs with loops and arrays using different data types (integers, floats, characters).

References

[1] B. Beizer, Software Testing Techniques, Van Nosterland Reinhold, 1990.

[2] J. A. Whittaker, "What is software testing? And Why is it so hard?" IEEE Software, 2000.

[3] B. T. de Abreu, E. Martins, F, de Sousa, "Automatic Test Data Generation for Path Testing Using A New Stochastic Algorithm", http://www.sbbd-sbes2005.ufu.br/arquivos/16-%209523.pdf

[4] B. Korel, "Automatic Software Test Data Generation", IEEE Trans of Software Engineering, 16(8):870-879.

[5] C. Michael, G. McGraw, M. Schatz, C. Walton, "Genetic Algorithm for Dynamic Test Data Generation", Proceedings of the 12th International Conference of Automated Software Engineering, vol.1, issue 5, Nov.1997, pp: 307-308.

[6] H. Sthamer, "The Automatic Generation of Software Test Data Using Genetic Algorithms", PhD. Thesis, University of Glamorgan, Wales, Great Britain, 1996.

[7] J. Wegener, K. Buhr, H. Pohlheim. "Automatic Test Data Generation for Structural Testing of Embedded Software System by Evolutionary Testing", http://www.lri.fr/~marc/articls/testwegener2002.pdf

[8] H. Sthamer, A. Baresel, J. Wegener, "Evolutionary Testing of Embedded Systems", 14th International Internet Quality week 2001.

[9] D. Berndt, J. Fisher, L. Johnson, J. Pinglikar, and A. Watkins, "Breeding Software Test Cases with Genetic Algorithms", IEEE Proceedings of the Hawaii International Conference on System Science, Hawaii, 2003.

[10] D. E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison Wesley, 1989.

[11] M.R .Girgis, "Automatic Test Data Generation for Data Flow Testing Using Genetic Algorithm", Journal of Universal Computer Science, vol. 11, no.6, 2005.

[12] D. Sandler, W. Tisthammer, "A Survey of Testing Tools", www.users.umn.edu/~dliang/5802reports/06/sandler/surveyt esting.pdf

[13] J. Lin, P.L.Yeh, "Automatic Test Data Generation of Path Testing Using GAs", Information Sciences, 131(1-4):47-64.

[14] M. S. Al-Zabidi, A. Kumar, A. D. Shaligram, "Data Generation for Path Testing Using Genetic Algorithms", In: Proceedings of the 1st International Engineering Sciences Conference (IESC'08), Aleppo, Syria, Nov. 2 - 4, 2008

[15] K. Borgelt, "Software Test Data Generation from a Genetic Algorithm", Industrial Applications of Genetic Algorithms, CRC Press, 1999.

[16] N. Mansour, M. Salame, "Data Generation for Path Testing". Software Quality Journal, 12,121-136-2004.

[17] G. Myer, The Art of Software Testing, John Wiley, 1979.

[18] R. P. Pargas, M. J. Harrold, R.Peck, "Test Data Generation Using Genetic Algorithms", Journal of Software Testing, Verification and Reliability, John Wiley, 1999.

Maha Al-Zabidi, received M.Sc. in Computer Science from University of Technology, Baghdad in 2000. She received B.SC. in Computer Science from King Abdul-Aziz University, Jeddah, KSA, in 1994. Since 2000 she has been working as an assistant lecturer in Hodeidah University, Yemen. She is presently doing a PhD. research at University of Pune, India. Her area of interest is software engineering, software quality, Genetic Algorithms and software testing.

Dr. Ajay Kumar, he has completed M.Sc. Engg. in computer science and Ph.D. in 1992. He is having 22 years of teaching and research experience. At present he is working as director of JSPM's Jaywant institute of computer applications, under University of Pune, Pune. He has published 36 research paper of national and international level. His areas of specialization are computer networks, mobile computing wireless networks, software engineering and data coding

Dr.A.D.Shaligram, received B.Sc. (1979), M.Sc. (1981) and Ph.D. (1986) from University of Pune. Presently he is head, Department of Electronic Science at University of Pune, has a professional experience of more than 25 years. His main fields of research interest are Optoelectronic sensors and systems, Wireless Sensor Networks, Simulation software development, Biomedical Instrumentation and sensors, PC/Microcontroller based instrumentation, Embedded systems and VLSI design, e-learning resource development. He has authored 20 text books and laboratory manuals for undergraduate electronics curricula, published more than 120 research papers in national/ international journals and conference proceedings. He has been a IEEE member for 16 years. He guided 18 students for Ph.D.