International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Impact Factor (2012): 3.358

Survey Paper on Optimum Selection of GA Algorithm's Parameters for Software Test Data Generation

Sonam Kamboj[1], Mohinder Singh[2]

[1]M. Tech Scholar, Department of CSE, Maharishi Vedvyas Engineering College, Haryana, India

[2]Professor CSE, Maharishi Vedvyas Engineering College, Haryana, India

Abstract: ***This paper empirically evaluates four metaheuristic search techniques namely Genetic Algorithm, artificial bee colony and Bing Bang Big Crunch Algorithm for automatic test data generation for procedure oriented programs using structural symbolic testing method. Test data is generated for each feasible path of the programs. All the four algorithms have been evaluated on average percentage coverage per path.***

Keywords: Generic Algorithm, Bing Bang Big Crunch Algorithm, symbolic testing method, average percentage coverage per path

1. Introduction

Although manual generation of test cases is relatively easy but it is a slow and costly process. Automatic generation of test cases can save time and testing resources. At the same time, it is also free from human biases and doesn't require special team of testers other than the developers. Despite having so many benefits, automated test case generation is not so easy because it requires intelligence of human mind to identify the nonlinearity and discreteness in test inputs' search space. For improving the quality of automation and fulfilling the requirements of test case generation, many researchers have explored new soft computing based techniques such as genetic algorithm, simulated annealing, tabu search, ant colony optimization, particle swarm optimization, memetic algorithms etc. to fulfill testing requirement and to generate suitable test cases automatically.

Two important concern of testing are:

- One is to select an optimum testing criterion also called as test adequacy criterion or test coverage criterion, which is feasible, effective and efficient to follow.
- Second is regarding automation of test case generation to avoid tester's biases and to reduce the huge cost involved in software testing.

1.1. 1 Software testing using population based approaches

Search techniques are applied for generation of test data by transforming testing objective into search problem. Two components are essential for a problem which is to be modeled as search target. First a mechanism should be derived through which the problem is encoded in search algorithm and second component is assessment of the suitability of solutions produced by search technique to guide the individuals for exploring search space. The population based metaheuristic search algorithm where global population represents every possible solutions and global search space, are frequently applied in applications where search space is very large. Each member of population is called an individual or a probable solution which is evaluated for its fitness so that new and better individual(s) may be generated.

1. Generate initial population comprising multiple individuals
2. Evaluate each individual following a criterion specific to problem
3. Generating next population by using current population based on their fitness.
4. Go to step 2 until stopping criterion is met

Figure 1: General Algorithm for a population based search algorithm

1.1.2 Fitness Function Design for symbolic path testing

In path testing approach a candidate solution (also called an individual) is used to evaluate constraint system of the target path. This evaluation can be dynamic as well as static. In dynamic analysis, a program is actually executed with values of the inputs and then fitness function determines the extent up to which it has satisfied the testing criterion, which becomes the fitness of the individual. On the other hand, static testing does not require the actual execution of program, but it symbolically executes a testing path as identified from CFG of program by using symbols instead of actual values. Symbols are replaced for variables in predicates or constraints of the entire target path and then this resultant constraint system is evaluated for fitness.

1.1.3 ABC algorithm

This is biologically inspired technique of swarm intelligence for searching. It is all about honey bees' work distribution and collective foraging strategy to accumulate extra nectar for their survival in winter season. Seeley investigated the behavior of bees in distributing their work to optimize the collection of nectar. Instead of initiating exploration by all bees, some dedicated explorer bees (scout bees) are appointed to explore the "profitability" of flower patches in the surrounding environment. This profitability accounts various parameters such as amount of nectar in flower patches, sugar contents in nectar, distance of flower patches

Licensed Under Creative Commons Attribution CC BY

from bee hive etc. If an explorer bee satisfies itself that there is sufficient profitability then it recruits unloader for unloading the nectar it has collected during exploration and dances (known as waggle dance) on dance floor (a designated place in beehive) to give feedback to foragers (observer or onlookers bees, which actually collect nectar from patches) about the quality of the flower patch, which they have recently searched out.

1.1.4 Fitness Function Design for path testing in symbolic execution environment

As explained in the previous section, search algorithm for testing generates population of candidate solutions which represent the test data taken from the search space modeled by the inputs' domains of the SUT. In path testing approach a candidate solution (also called an individual) is used to evaluate constraint system of the target path. This evaluation can be dynamic as well as static. In dynamic analysis, a program is actually executed with values of the inputs and then fitness function determines the extent up to which it has satisfied the testing criterion, which becomes the fitness of the individual. On the other hand, static testing does not require the actual execution of program, but it symbolically executes a testing path as identified from CFG of program by using symbols instead of actual values. Symbols are replaced for variables in predicates or constraints of the entire target path and then this resultant constraint system is evaluated for fitness.

Violated individual predicate Penalty to be imposed in case predicate is not satisfied

A < B A – B + ζ
A <= B A – B
A > B B – A + ζ
A >= B B – A
A = B Abs(A – B)
A ≠ B ζ – abs(A – B)

A and B are operands and ζ is a smallest constant of operands' universal domains. In case integer it is 1 and in case real values it can be 0.1 or 0.01 depending on the accuracy we need in solution.

1.2 GA for software testing

Genetic algorithms (GAs) are optimization technique initially inspired from the processes of natural selection and is considered good for searching nonlinear and discrete search spaces. GA starts with an initial population in which each individual member (henceforth called individual) is called chromosome or candidate solution and elements of each chromosome are called genes. Subsequently, it uses genetic operators: selection, crossover and mutation on fitter individuals iteratively in order to generate next population, which has the high probability of fulfilling test coverage or adequacy criterion in larger degree as compared to its parents. Each individual in GA population is analyzed for its fitness. Being a population based search algorithm, GA's success in achieving objectives largely depends on the meritorious definition of fitness function.

1.3 PSO Algorithm for Software Testing

PSO is a biologically inspired algorithm which applies to concept of social interaction to problem solving. In PSO a swarm of "n" individuals or particles (starting population generated randomly like GA) communicates either directly or indirectly with one another to generate the next better search options. Each particle "flies" in the direction of a better solution weighted by some random factor, sometime overshooting or another time finding a better or globally better position. Each particle records and updates its best fitness and corresponding position in successive iterations. Global fitness (best fitness among all particles over all generation) and corresponding position is also remembered. The interaction between the particles in the swarm helps to prevent staying off, while keeping close to the optimal solution. This type of behavior is ideal when exploring large search spaces.

2. Literature Review

Premal B. Nirpal and K. V. Kale [1] The algorithm takes a selected path as a target and executes sequences of operators iteratively for test cases to evolve. The evolved test case can lead the program execution to achieve the target path. An automatic path-oriented test data generation is not only a crucial problem but also a hot issue in the research area of software testing today.

Jitender kumar chhabra [2] Test cases are symbolically generated by measuring fitness of individuals with the help of branch distance based objective function. Evaluation of the test generator was performed using ten real world programs. Some of these programs had large ranges for input variables. Results show that the new technique is a reasonable alternative for test data generation, but doesn't perform very well for large inputs and where constraints are having many equality constraints.

Parveen kumar [3] Genetic Algorithm and Big Bang Big Crunch Algorithm for automatic test data generation for procedure oriented programs using structural symbolic testing method. Test data is generated for each feasible path of the programs. Experiments on ten benchmark programs of varying sizes and complexities are conducted and the subsequent performance results are presented. All the four algorithms have been evaluated on average test cases per path and average percentage coverage per path. It has been observed that the particle swarm optimization based algorithm outperforms the other three algorithms. The result also concludes that predicates solving difficulty (such as constraints having equality operator'&&' as join operator) has a direct relationship with testing efforts rather than program complexity measures such as cyclomatic complexity, number of decision nodes etc.

Shakti kumar [4] Software engineers are facing uphill task of stabilizing software testing cost with ever increasing software complexity. Software test data generation is tedious, most time consuming, complex and central activity of testing. But this is also the only task in testing where automation can be deployed. Besides being NP-hard problem, software testing also requires exact solution

Licensed Under Creative Commons Attribution CC BY

making it more demanding as compared to other optimization problems. With this background, Researchers are enthusiastically seeking employment of heuristic methods towards test data generation. This paper compares and evaluates two swarm intelligence based search techniques namely particle swarm optimization and artificial bee colony algorithm for automatic test data generation for procedure oriented programs using structural symbolic testing method. Test data is generated for each feasible path of the programs. Experiments on ten real world programs of varying sizes and complexities are conducted and the subsequent performance results are presented. The results of these approaches are also compared with genetic algorithm based technique for test data generation for demonstrating the efficiency of the swarm intelligence algorithms. The three algorithms have been evaluated on average test cases per path and average percentage coverage per path. It has been observed that the particle swarm optimization based algorithm outperforms the other two algorithms.

Harmen - Hinrich Sthamer [5] Random testing is used as a comparison of the effectiveness of test data generation using GAs which requires up to two orders of magnitude fewer tests than random testing and achieves 100% branch coverage. The advantage of GAs is that through the search and optimization process, test sets are improved such that they are at or close to the input sub domain boundaries. The GAs give most improvements over random testing when these sub domains are small. Mutation analysis is used to establish the quality of test data generation and the strengths and weaknesses of the test data generation strategy.

Phil McMinn [6] The use of metaheuristic search techniques for the automatic generation of test data has been a burgeoning interest for many researchers in recent years. Previous attempts to automate the test generation process have been limited, having been constrained by the size and complexity of software, and the basic fact that in general, test data generation is an undecidable problem. Metaheuristic search techniques o_er much promise in regard to these problems. Metaheuristic search techniques are high level frameworks, which utilize heuristics to seek solutions for combinatorial problems at a reasonable computational cost. To date, metaheuristic search techniques have been applied to automate test data generation for structural and functional testing; the testing of grey-box properties, for example safety constraints; and also non-functional properties, such as worst-case execution time. This paper surveys some of the work undertaken in this field, discussing possible new future directions of research for each of its different individual areas.

3. Conclusion

In this survey detail study of test data generation various testing algorithms have been done. For generation of test cases, symbolic execution method has been used in which first, target path is selected from CFG of SUT and then inputs are generated using search algorithms which can evaluate composite predicate corresponding to the target path true.

References

[1] Ahmed MA and Hermadi I, GA-based multiple paths test data generator. *Computers and Operations Research* (2007),(article in press)

[2] Alba E and Chicano F, Software Project Management with Gas. *Information Sciences*, 2007; 177(11):2380-2401

[3] Amoui M, Mirarab S, Ansari A and Lucas C, A Genetic Algorithm Approach to Design Evolution using Design Pattern Transformation. *International Journal of Information Technology and Intelligent Computing* 1(2), 2006; 235-244.

[4] Ayari K, Bouktif S and Antoniol G, Automatic Mutation Test Input Data Generation via Ant Colony. *GECCO'07*, July 7–11, 2007, London, England, United Kingdom.

[5] Beizer B. *Software testing techniques*. 2nd ed., Dreamtech publication New Delhi. 1990.

[6] Burgess CJ and Lefley M, Can Genetic Programming Improve Software Effort Estimation? A Comparative Evaluation. *Information & Software Technology*, 2001; 43(14):863-873

[7] Chong CS, Low MYH, Sivakumar AI and Gay KL, A Bee Colony Optimization Algorithm to Job Shop Scheduling. *Proceedings of the 37th Winter Simulation*, Monterey, California, 1954-1961, 2006.

[8] Clow B and White T, An evolutionary race: A comparison of genetic algorithms and particle swarm optimization for training neural networks. *In Proceedings of the International Conference on Artificial Intelligence, IC-AI '04*, Volume 2, pages 582–588. CSREA Press, 2004.

[9] Dahiya SS, Chhabra JK and Kumar S, Application of Particle Swarm Optimization Algorithm to Symbolic Software Testing. *ADCOM 2009*, to be held in Bangalore on 14-17 December 2009. (Communicated for publication)

[10] Demillo RA, and Offutt AJ, Constraint-based automatic test data generation. *IEEE transaction on Software engineering*. 1991; 17(9): 900-910

[11] DeMillo RA, Lipton RJ and Sayward FG, Hints on Test Data Selection: Help for the Practicing Programmer. *IEEE Computer*, 1978; II(4): 34-41.

[12] Díaz E, Javier T, Raquel B and José JD, A tabu search algorithm for structural software testing. *Computers and Operations Research* (2007), doi:10.1016/j.cor.2007.01.009

[13] Duran JW and Ntafos AS, Report On Random Testing. *International Conference on Software engineering,* San Diego, California, United States *March 09 - 12, 1981*

[14] Edvardsson J, A survey on automatic test data generation. *In Proceedings of the second conference on computer science and engineering*, Linkoping: ESCEL; October 1999; 21–28.

[15] Frankl PG and Weyuker EJ, An Applicable Family of Data Flow Testing Criteria. *IEEE Transaction on Software Engineering.* 1988; 14(10):1483-1498.

[16] Gilb T and Graham D. *Software Inspection*, Addison-Wesley 1993

Licensed Under Creative Commons Attribution CC BY

[17] Goldberg DE. *Genetic algorithms in search, optimization, and machine learning,* Addison-Wesley, 1989.
[18] Harman M and Jones BF, Search-based Software Engineering. *Information & Software Technology*, 2001; 43(14):833-839.
[19] Huaizhong LI and LAM Peng C, An Ant Colony Optimization Approach to Test Sequence Generation for State based Software Testing. *Proceedings of the Fifth International IEEE Conference on Quality Software (QSIC'05)* 2005.
[20] Jones KO, Comparison of genetic algorithm and particle swarm optimization. *In Proceedings of the International Conference on Computer Systems and Technologies*, 2005.
[21] Jorgenson P. *Software Testing: A Craftman's Approach*, 2nd edition CRC Press, Inc. Boca Raton, FL, USA, 2001.
[22] Korel B, Automated software test data generation. *IEEE transaction on software engineering*, 1990; 16(8):870-879.
[23] Laitenberger O and DeBaud J, An encompassing life cycle-centric survey of software inspection. *Journal of System Software,*. 50 (2000):5–31.
[24] Lin JC and Yeh PL, Automatic test data generation for path testing using GAs. *Information Sciences* 2001; 131:47–64.
[25] Mansour N and Salame M, Data generation for path testing. *Software Quality Journal* 2004; 12:121–136.
[26] Mantere T and Alander JT, Evolutionary Software Engineering, A Review, *Applied Soft Computing,* 2005; 5(3):315-331
[27] Mayer J, Schneckenburger C, An Empirical Analysis and Comparison of Random Testing Techniques, *ISESE'06,*September 21–22, 2006, Rio de Janeiro, Brazil pp. 105-114.
[28] McMinn P. Search-based Software Test Data Generation: A Survey. *Software Testing, Verification and Reliability* June 2004; 14(2):105-156.
[29] Michael C, McGraw G and Schatz M. Generating software test data by evolution. *IEEE Transactions on Software Engineering* 2001; 27(12):1085–1110.
[30] Miller W and Spooner D. Automatic generation of floating-point test data. *IEEE Transactions on Software Engineering* 1976; 2(3):223-226.
[31] Mitchell BS and Mancoridis S, On the Automatic Modularization of Software Systems using the Bunch Tool. *IEEE Transactions on Software Engineering*, 2006; 32(3):193-208
[32] Myers GJ. *The art of software testing*. New York: Wiley; 1979
[33] Nakrani S and Tovey C, On Honey Bees and Dynamic Allocation in an Internet Server Colony. *Proceedings of 2nd International Workshop on the Mathematics and Algorithms of Social Insects*, Atlanta, Georgia, USA, 2003.
[34] Pargas RP, Harrold MJ and Peck R, Test-data generation using genetic algorithms. *Journal of Software Testing, Verification and Reliability* 1999; 9(4):263–82.
[35] Pedrycz W, Computational Intelligence as an Emerging Paradigm of Software Engineering. *Proceedings of the 14th International ACM Conference on Software Engineering and Knowledge Engineering (SEKE '02),* 2002, pp. 7-14
[36] Pham DT, Otri S, Afify A, Mahmuddin M, and Al-Jabbouli H, Data clustering using the Bees Algorithm. *In 40th CIRP International Seminar on Manufacturing Systems*. 2007: Liverpool.
[37] Pham DT, Otri S, Ghanbarzadeh A, Kog E, Application of the Bees Algorithm to the Training of Learning Vector Quantisation Networks for Control Chart Pattern Recognition. *ICTTA'06 Information and Communication Technologies*, 1624-1629, 2006b.
[38] Porter A, Sey A and Votta L, A review of software inspections. *Technical Report: CS-TR-3552*, University of Maryland at College Park College Park, MD, USA, 1995
[39] Roper M, Computer aided software testing using genetic algorithms. In 10th *International Software Quality Week*, San Francisco, USA, 1997.
[40] Schmickl T, Thenius R and Crailsheim K, Simulating Swarm Intelligence in Honey Bees: Foraging in Differently Fluctuating Environments, *GECCO'05*, Washington, DC, USA, 273-274,2005.
[41] Seeley TD, *The Wisdom of the Hive*, Harvard University Press, Cambridge, MA, 1995.

Licensed Under Creative Commons Attribution CC BY