Examining Search-Based Algorithms in Software Testing

[1]Sukhmanpreet Kaur, [2]Parminder Kaur

Department of Computer Science & Engineering, Guru Nanak Dev University, Amritsar, India
Email: [1]sukhmanpreetkaur41@ymail.com, [2]parminderkaur@yahoo.com

Abstract—Software testing is a way of improving software quality. It is an essential and an expensive phase of Software Development Life Cycle. There has been an ongoing research in this field to automate the process of software testing so that expenses can be reduced. But size and complexity of software pose hindrance in their automation. Meta-heuristic and evolutionary algorithms have proved to be much useful for automating the process of test generation. Usages of meta-heuristic approaches have led to the emergence of new field in software engineering. This field is known as Search-Based Software Engineering (SBSE). SBSE is applicable to wide range of software engineering problems. Application of these approaches to software testing has come to be known as Search-Based Software Testing. This paper examines several search-based algorithms. These algorithms are compared to one another on the basis of various parameters taken into consideration. All of these algorithms are strongly dependent on problem domain as heuristics related to that domain are very much essential for carrying out execution of the problem using desired algorithm.

Index Terms—Software Testing, Search-Based Software Engineering, Search-Based Software Testing, Genetic Algorithms

I. INTRODUCTION

Software testing is the process of evaluating the quality of the developed software by finding as many faults as possible. It is an important phase of software development lifecycle which alone accounts for 40% to 50% of software development cost and this cost may vary with size and other parameters related to the chosen project. Automated testing is essential for modern complex software systems as the cost of manual testing is very high. From the last few decades, there have been constant attempts to reduce the time and efforts required for software testing by automating the process of software test data generation.

The last decade has witnessed much research in applying search-based optimization methods to this problem. This area of search is known Search-Based Software Testing (SBST). This is an instance of Search-Based Software Engineering (SBSE). This term was given by Harman and Jones in 2001. SBSE consists of the use of search based optimization algorithms such as hill climbing, simulated annealing, and genetic algorithms being the most commonly used in the field of software engineering. These search algorithms are attractive in software engineering due to the reason that data are often inaccurate, incomplete and dispersed over larger area which makes traditional optimization techniques incompatible with the given data. SBST deals with a testing task by automating it with the help of meta-heuristic algorithms. Wide range of algorithms can be used for this purpose. All these algorithms are dependent on problem domain. No matter which algorithm is used, it is the fitness function that guides the search and captures the crucial information and differentiates a good solution from a poor one.

II. LITERATURE SURVEY

Search-Based Software Engineering

This field has gained much popularity in the last decade. It involves the use of search based optimization techniques for various software engineering activities during the lifecycle of software, such as project planning, cost estimation [1,2,3,11 ,15], requirement engineering [4], testing [5,6,7,9,10,16,17,20,21,22], automated maintenance [11,13,14,19,23,24,25], quality assessment [8,18], etc. When problems in software engineering are solved using search-based software engineering techniques, they show improvements in results and therefore this field has proved to be very much beneficial for wide range of software engineering problems. The search-based algorithms follow these basic steps:

- Search Initialization: The search is initiated by randomly choosing solution from possible candidate solutions.
- Quality Assessment: Assessing the quality of a candidate solution by means of fitness function.
- Modify: Modifying the candidate solution by making it slightly different.
- Select: Selecting the candidate solution on the basis of fitness function in accordance with chosen algorithm.

Categorizing search-based algorithms [28]:

- Local or Global search algorithms
- Single state or Population based algorithms
- Local and Global optimization algorithms: Local search algorithms find local optima and need to restart again from a different point in order to obtain global optima whereas in case of global search algorithms, local optima are avoided. There is a trade-off between local and global search algorithms. Global search algorithms have higher efficiency but require greater cost and effort for computation. Local search algorithms are more effective for simple problems.

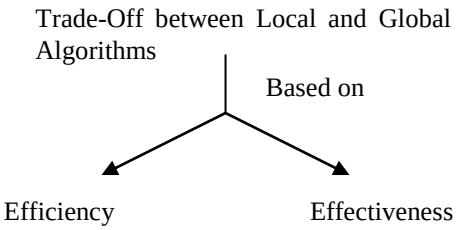

Table1. Local and Global Search Algorithms

Local Search Algorithm	Global Search Algorithm
Deals with simple problems	Deals with complex problems
Not very efficient	Have higher efficiency
Require less effort and cost for computation	Require greater cost and effort for computation

- Single state or Population based algorithms:

Single state methods find one solution at a time whereas in population based methods, many candidate solutions are used at a time. Population based methods are based on evolutionary algorithms. Therefore they need modify step which involves mutation and recombination of fittest parents to create even better children.

Table2. Single state or Population based Algorithms

Single State Methods	Population State Methods
Evaluate one candidate solution at a time.	Evaluate many candidate solutions at a time.
Process of neighborhood evaluation is used here.	Process of crossover and mutation are used here.
e.g. Hill Climbing, Simulated Annealing	e.g. Genetic Algorithms

A. Search-Based Software Testing (SBST) [26, 27, 28]

Search-Based Software Testing has proved to be very useful for software testing. It helps to automate a testing problem. It involves the use of SBSE optimization Algorithms [2, 3, 4, 1]. Here also fitness function plays a very important role in finding solution to a problem. All algorithms are problem dependent as heuristics based on problem domain are used for evaluating the solution to a problem. Algorithm to be used for a problem is dependent on the type of problem taken into consideration.

Generally there are two rudimentary requirements that need to be fulfilled in order to apply search- based optimization techniques to a testing problem.

- Representation: - The problem needs to be represented so that it can be manipulated by the search algorithm.
- Fitness Function: - The function for guiding the search. It is problem specific. It is this function that carries crucial information regarding the problem and it helps to distinguish between good and poor solution.

B. Search-Based Optimization Algorithms [6, 7, 8, 9, 10, 11, 12, 26, 27, 28]

Hill Climbing:

It is one of the simplest search based optimization algorithm. It is effective for simple problems. It is a local search algorithm which starts from a randomly chosen candidate solution. At each step, the neighbors of candidate solution are evaluated for fitness. If a better candidate solution is found, move is made to that neighbor else it is discarded. This way the process is continued till no fitter neighbor is left. Then the search is terminated. If local optima are reached, then the same process is restarted from a new randomly chosen point till global optima are reached.

Simulated Annealing:

This algorithm is inspired from the chemical process of annealing. Annealing refers to slow cooling of highly heated material. The properties of cooled material depend at the rate at which the cooling takes place. It is similar to hill climbing in a way that neighbor is considered for better fitness but it allows probabilistic moves to poorer solutions to avoid local maxima. Initially when the temperature is high, free movement around the search space is allowed so that the search is less dependent on the starting solution. As the search advances, the temperature decreases and there is less freedom of movement. If the cooling is too fast, enough search space will not be explored and there are more chances of obtaining local maxima.

Genetic Algorithms:

They are also known as evolutionary algorithms. They are inspired from the evolutionary process of biology. They are population based search algorithms that involve the process of natural evolution (mutation and crossover) for selecting the fittest individual. These algorithms have immense applications in the field of search based

software testing. They involve natural process of evolution. It starts with a random generation of initial population. The individuals of the population are represented by chromosomes and they are the encoded solutions to a problem. These chromosomes again undergo evolution on basis of certain rules, mutation and reproduction. Evaluation of fitness for each individual takes place. Again parents recombine to form new off springs and the process continues until the solution to the problem has been found or stopping condition is reached. The stopping Condition may depend on available number of resources or the maximum number of iterations.

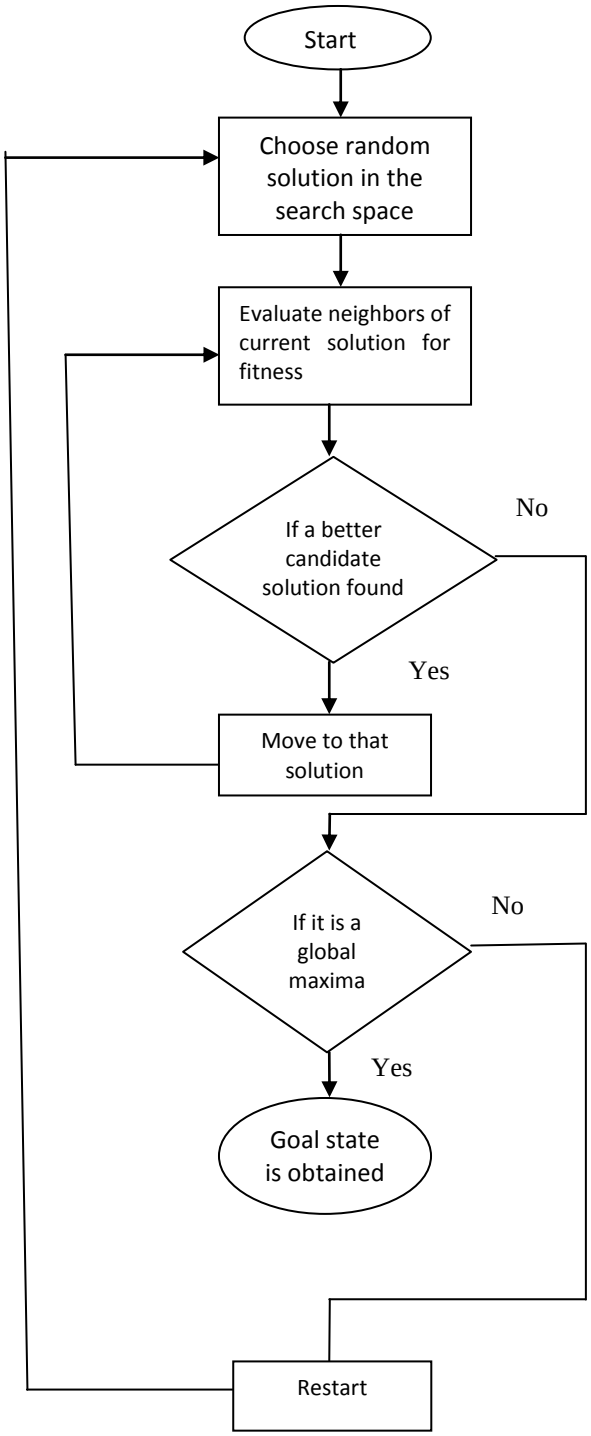

Fig1.Flowchart for Hill Climbing Algorithm

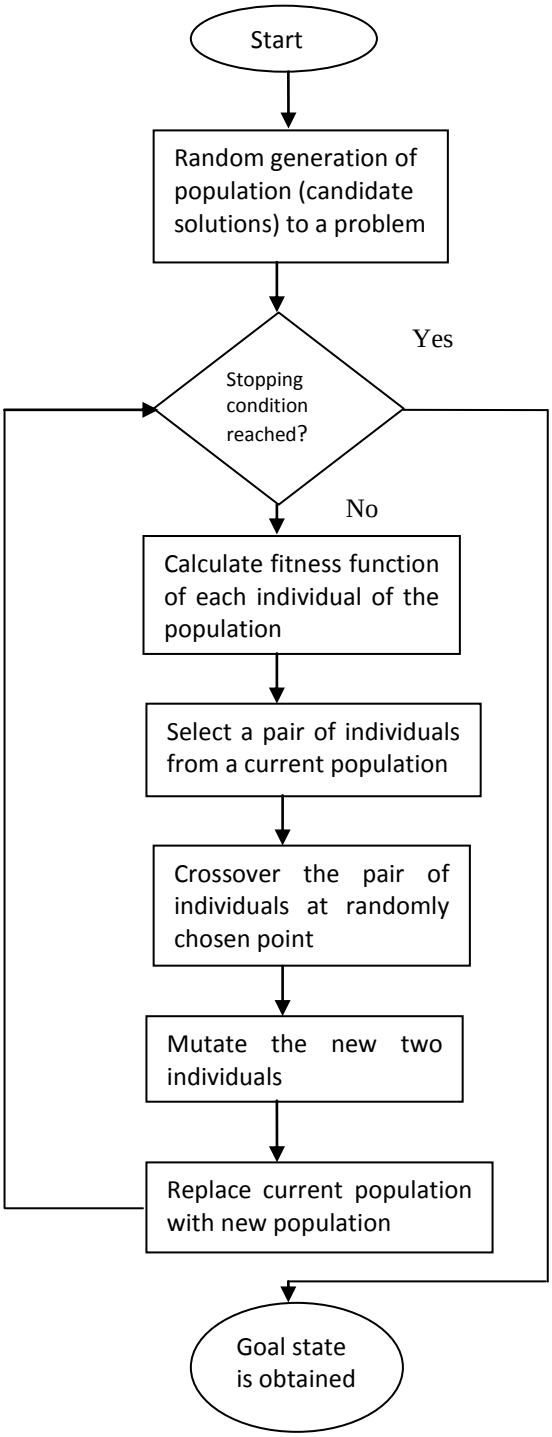

Fig2. Flowchart for Genetic Algorithm

III. INFERENCE

The three meta-heuristic algorithms that have been used in software testing are compared in this section. As we know that the Search-Based-Software-Testing algorithms are strongly dependent on the domain of the problem so each algorithm has its own applications where they can be used more efficiently. This paper has reviewed some common search algorithms. The various algorithms that have been reviewed are hill climbing, simulated annealing, and genetic algorithms. In this section some

crucial parameters are taken into consideration and on the basis of these parameters, an algorithm is chosen that can be used in a given situation. The various parameters taken into consideration are:

- Meta-heuristic approach
- domain specific
- local search approach
- global search approach
- search one point at a time
- fitness function used to guide search
- backtracking used to deal with local maxima
- neighborhood dependence
- principle of crossover, mutation used
- simplicity

Each algorithm is characterized by its special features. A particular algorithm cannot be used in every situation. One algorithm is chosen based on our requirements. Requirement specifications while choosing an algorithm is very important. So we can say that each algorithm has its own advantages and disadvantages so the chosen problem will define which algorithm will be appropriate for a given problem (table 3, figure 3). Hill climbing is a local search approach. On the other hand simulated annealing and genetic algorithms are global search algorithms, finding many solutions in the search space at a given time. In each algorithm, fitness function is essential as it helps in guiding the search.

IV. FUTURE CHALLENGES

Future Challenges for meta-heuristic Algorithms

C. Stopping Criteria [26, 27]:

To terminate search algorithms, certain stopping criterion is required. Much of the previous work has adopted one of the following two approaches to terminate the search.

- They are taken to be some time or budget constraint on effort required for computation.
- It may act as a criterion that must be met by the proposed solution.

But in case of evolutionary algorithms, there is a third possibility i.e. the search is terminated when all the individuals of a population become homogeneous. In this case, when the individuals have similar chromosomes there is very little chance of further improvements in fitness. So, a question is raised that how can we measure similarity among solutions. This is domain specific. Therefore certain metrics are required that can help in measuring the similarity of a set of candidate solutions for wide range of software engineering problems. These metrics need to be cost effective as they will be required at regular intervals during the search.

D. Memetic Algorithms [26, 27]:

This is an algorithm that is used in SBST that combines the features of other SBST algorithms such as hill climbing, Simulated Annealing, genetic Algorithms. Therefore it can be called hybrid technique in SBST. It takes into consideration best aspects of local and global search. A simple example of memetic algorithm can be of a genetic algorithm that takes a stage of hill climbing at the end of each generation to improve the quality of each individual of the population to a certain extent. They are best suited for problems with unpredictable landscape. Applications of these techniques need to be investigated in structural test data generation problem.

E. Fitness Landscape Visualization [26, 27,]:

It is used to visualize an optimization problem at hand by means of distributing fitness values of the candidate solutions in the search space. When each individual occupies location on the horizontal plane, the landscape is visualized by the use of fitness function values as a measure of height in the landscape. Here the best solution to a problem is represented by the highest point in the landscape. The shape of the visualization landscape affects the progress of the search. If the landscape is free of local optima, the search will be quite easy. On the other hand if search problem has a complex landscape containing several optima then the search may be either misled or offered little guidance. So fitness landscape visualization helps to determine which search technique will be best suited for the problem at hand.

Table3. Comparison among Meta-Heuristic Search Algorithms

ALGORITHMS / PARAMETERS	HILL CLIMBING	SIMULATED ANNEALING	GENETIC ALGORITHM
Meta-heuristic Approach	✓	✓	✓
Domain specific	✓	✓	✓
Local Search Approach	✓	×	×
Global Search Approach	×	✓	✓
Search one solution at a time	✓	✓	×
Fitness function	✓	✓	✓
Simplicity	✓	×	×
Neighborhood Dependent	✓	✓	×
Principle of mutation used	×	×	✓
Backtracking used to deal with local maxima	✓	×	×

V. CONCLUSIONS

As we all know that the cost of manual testing in practice is very high, therefore research into automated software testing is very generic approach in which solutions may be sought for various software testing problems automatically using optimization algorithms. This paper has reviewed some common search based algorithms like hill climbing, simulated annealing, and genetic algorithms. Each of these algorithms has its own advantages and disadvantages compared to other algorithms. Hill Climbing algorithms are known as local search approaches because they consider only one solution at a time. This approach is simple but sometimes inefficient and time consuming as they move only in the local neighborhood of those solutions. They could not escape from local optima in the search space of possible input data. To overcome this problem backtracking to some earlier solution is required. Simulated Annealing Algorithms are similar to hill climbing in a way that they consider one solution at a time and move in local neighborhood of those solutions. However they allow probabilistic moves to poorer solutions to avoid local maxima. On the other hand Genetic Algorithms are a form of global search, sampling many solutions at a time. In the last two decades genetic algorithms have been widely employed for various test data generation.

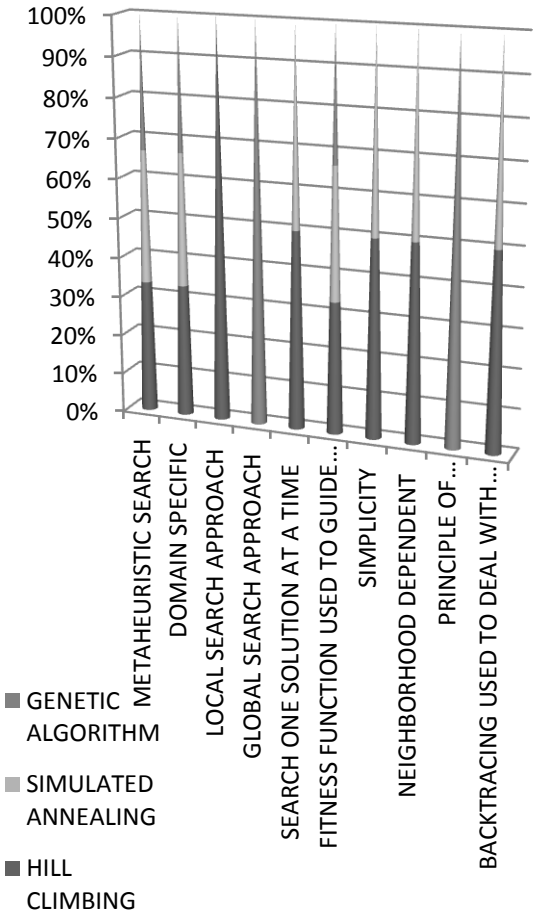

Fig3. Algorithms satisfying various parameters

REFERENCES

[1] J. Aguilar-Ruiz, I. Ramos, J. C. Riquelme, and M. Toro.“An evolutionary approach to estimating software development projects” Information and Software Technology,43(14):875–882, Dec. 2001.

[2] G. Antoniol, M. Di Penta, and M. Harman. “A robust search–based approach to project management in the presence of abandonment, rework, error and uncertainty” In 10th International Software Metrics Symposium (METRICS 2004),pages 172–183, Los Alamitos, California, USA, Sept. 2004.IEEE Computer Society Press.

[3] G. Antoniol, M. D. Penta, and M. Harman. “Search-based techniques applied to optimization of project planning for amassive maintenance project”. In 21st IEEE International Conference on Software Maintenance, pages 240–249, LosAlamitos, California, USA, 2005. IEEE Computer Society.

[4] A. Bagnall, V. Rayward-Smith, and I. Whittley.”The next release problem”. Information and Software Technology,43(14):883–890, Dec. 2001.

[5] A. Baresel, D. W. Binkley, M. Harman, and B. Korel. Evolutionary testing in the presence of loop–assigned flags: A testability transformation approach. In International Symposium on Software Testing and Analysis (ISSTA 2004), pages108–118, Omni Parker House Hotel, Boston, Massachusetts, July 2004. Appears in Software Engineering Notes, Volume29, Number 4.

[6] A. Baresel, H. Sthamer, and M. Schmidt. Fitness function design to improve evolutionary structural testing. In GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference, pages 1329–1336, San Fran- Future of Software Engineering (FOSE'07) 0-7695-2829-5/07 $20.00 © 2007 cisco, CA 94104, USA, 9-13 July 2002.Morgan Kaufman Publishers.

[7] L. Bottaci. Instrumenting programs with flag variables for test data search by genetic algorithms. In GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference, pages 1337–1342, New York, 9-13 July 2002.

[8] S. Bouktif, H. Sahraoui, and G. Antoniol. Simulated annealing for improving software quality prediction. In GECCO2006: Proceedings of the 8th annual conference on Genetic and evolutionary computation, volume 2, pages

1893–1900,Seattle, Washington, USA, 8-12 July 2006. ACM Press.

[9] L. C. Briand, J. Feng, and Y. Labiche. Using genetic algorithms and coupling measures to devise optimal integration test orders. In SEKE, pages 43–50, 2002.

[10] L. C. Briand, Y. Labiche, and M. Shousha. Stress testingreal-time systems with genetic algorithms. In Genetic Computation Conference, GECCO 2005, Proceedings, Washington DC, USA, June 25-29, 2005, pages1021–1028. ACM, 2005.

[11] C. J. Burgess and M. Lefley. Can genetic programming improve software effort estimation? A comparative evaluation. Information and Software Technology, 43(14):863–873, Dec. 2001.

[12] D. Fatiregun, M. Harman, and R. Hierons. Search-based amorphous slicing. In 12th International Working Conference on Reverse Engineering (WCRE 05), pages 3–12, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA, Nov. 2005.

[13] M. Harman, R. Hierons, and M. Proctor. A new representation and crossover operator for search-based optimization of software modularization. In GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference, pages 1351–1358, San Francisco, CA 94104, USA, 9-13July 2002. Morgan Kaufmann Publishers.

[14] B. S. Mitchell and S. Mancoridis. Using heuristic search techniques to extract design abstractions from source code. In GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference, pages 1375–1382, San Francisco, CA 94104, USA, 9-13 July 2002. Morgan Kaufmann Publishers.

[15] J. J. Dolado. A validation of the component-based method for software size estimation. IEEE Transactions on Software Engineering, 26(10):1006–1021, 2000.

[16] Q. Guo, R. M. Hierons, M. Harman, and K. Derderian. Constructing multiple unique input/output sequences using evolutionary optimization techniques. IEE Proceedings — Software,152(3):127–140, 2005.

[17] M. Harman, L. Hu, R. M. Hierons, J. Wegener, H. Sthamer, A. Baresel, and M. Roper. Testability transformation. IEEE Transactions on Software Engineering, 30(1):3–16, Jan.2004.

[18] T. M. Khoshgoftaar, L. Yi, and N. Seliya. A multi objective module-order model for software quality enhancement. IEEE Transactions on Evolutionary Computation, 8(6):593–608, December 2004.

[19] B. S. Mitchell and S. Mancoridis. On the automatic modularization of software systems using the bunch tool. IEEE Transactions on Software Engineering, 32(3):193–208, 2006.

[20] Z. Li,M. Harman, and R. Hierons. Meta-heuristic search algorithms for regression test case prioritization. IEEE Transactions on Software Engineering, 2007, pages 225-237.

[21] P. McMinn, M. Harman, D. Binkley, and P. Tonella. Thespecies per path approach to search-based test data generation. In International Symposium on Software Testing and Analysis (ISSTA 06), pages 13–24, Portland, Maine, USA,2006.

[22] J. Wegener, A. Baresel, and H. Sthamer. Evolutionary test environment for automatic structural testing. Information and Software Technology Special Issue on Software Engineering using Meta-heuristic Innovative Algorithms,43(14):841–854, 2001.

[23] M. O'Keeffe and M. O'Cinneide. Search-based software maintenance. In Conference on Software Maintenance and Reengineering (CSMR'06), pages 249–260, Mar. 2006.

[24] O. Seng, M. Bauer, M. Biehl, and G. Pache. Search based improvement of subsystem decompositions. In H.-G.Beyer and U.-M. O'Reilly, editors, Genetic and Evolutionary Computation Conference, GECCO 2005, Proceedings, Washington DC, USA, June 25-29, 2005, pages 1045–1051.ACM, 2005.

[25] O. Seng, J. Stammel, and D. Burkhart. Search-based determination of refactorings for improving the class structure of object-oriented systems. In GECCO 2006: Proceedings ofthe 8th annual conference on Genetic and evolutionary computation, volume 2, pages 1909–1916, Seattle, Washington, USA, 8-12 July 2006. ACM Press.

[26] Sapna Varshney, Dr. Monica Mehrotra "Search based Software Test Data Generation for Structural Testing: A Perspective" ACM SIGSOFT July 2013.

[27] Mark Harman, "The Current State and Future of Search Based Software Engineering" Future of Software Engineering (FOSE'07) 2007.

[28] Phil McMinn, "Search-Based Software Testing: Past, Present and Future" University of Sheffield, Department of Computer Science Regent Court, 211 Portobello, and Sheffield, S1 4DP, UK

◈◈◈