iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 2 **Issue:** III **Month of publication:** March 2014

DOI:

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

ISSN: 2321-9653

INTERNATIONAL JOURNAL FOR RESEARCH IN APPLIED SCIENCE AND ENGINEERING TECHNOLOGY (IJRASET)

An Evolutionary Algorithm for Automated Test Data Generation for Software Programs

Preeti, Kompal Ahuja(Assistant Professor)

DCRUST, Murthal, Haryana (INDIA)

DITM, Gannaur, Haryana (INDIA)

Abstract: Software testing takes a considerable amount of time and resources spent on producing software it would be useful to have ways to reduce the cost of software testing so one of them is automatic test data generator- a system that automatically generates test data for a given program. This paper presents a Big Bang-Big Crunch concept based search algorithm for automatic generation of structural software tests. Big Bang-Big Crunch algorithm is a new meta-heuristic population based algorithm that relies on one of the theories of evolution of universe, namely, the Big Bang-Big Crunch theory.

Keywords: Big Bang-Big Crunch algorithm, Software Testing, Path Generator, Automatic test data generator, Fitness function

I. Introduction

Software testing is a main method for improving the quality and increasing the reliability of software. It is a kind of complex, labour-intensive and time consuming work. Software testing accounts for 50% of the total cost of software development. This cost could be reduced if the process of testing is automated. Automatic test data generator – a system that automatically generates test data for a given program. Test data generation in software testing is the process of program input data, which satisfy a given testing criterion. A meta-heuristic is a higher level procedure or heuristic designed to find, generate or select a lower level procedure or heuristic that may provide a sufficiently good solution to an optimization problem. Big Bang and Big Crunch algorithm is a new meta-heuristic population based algorithm that relies on one of the theories of the universe, namely the Big Bang and Big Crunch theory.

II. Meta-heuristic Search Algorithm

A meta-heuristic is a higher level procedure or heuristic designed to find, generate or select a lower level procedure or heuristic that may provide a sufficiently good solution to an optimization problem. Meta-heuristics are generally applied to problems for which there is no satisfactory problem-specific algorithm or heuristic; or when it is not practical to implement such a method. Most commonly used Meta-heuristics are targeted to combinatorial optimization problems, but of course can handle any problem that can be recast in that form, such as solving Boolean equations.

Soft computing and heuristic based techniques such as evolutionary algorithms, fuzzy modelling, neural networks and swarm intelligence based algorithms have found several applications in software engineering domain. Some of these applications are cost/effort estimation for better resource utilization and allocation in software development [Burgess2001, Shukla2000], automated software testing for cost cutting and making software more reliable [Michael2001], module clustering for effective maintenance [Mitchell2006] and software project management activities such as resource scheduling [Alba2007]. Considerable effort has been put in by various researchers to apply soft computing in the area of software engineering [Harman2001, Mantere2005]. One of the most intensively applied areas of soft computing in software engineering is test data generation, which is categorized as NP-hard [Yuan2005] or NP-complete [Mansour2004] problem due to requirement of enormous efforts in finding the data out of large search space satisfying the complex and non-linear search objectives. Some of the meta-heuristic search techniques which we have employed for

our experimentation are described below with their respective algorithm workflow.

A. Big Bang and Big Crunch Search Algorithm

Big Bang and Big Crunch algorithm is a new meta-heuristic population based algorithm that relies on one of the theories of the evolution of universe, namely the Big Bang-Big Crunch theory. The Big Bang and Big Crunch theory is introduced by Erol and Eksin [Erol2006], which is based upon the analogy of universe evolution. It has two phases viz 1. Big Bang Phase and 2. Big Crunch Phase.

In Big Bang phase, candidate solutions are randomly distributed over the search space and in the Big Crunch phase, randomly distributed particles are drawn into an orderly fashion. The Big Bang-Big Crunch optimization method generates random points in the Big Bang phase and shrinks these points to a single point in the Big Crunch phase. The Big Crunch phase has a convergence operator that has many inputs but only one output, which is named as the "centre of mass", since the only output has been derived by calculating the centre of mass. This concept can be mathematically simulated by obtaining object function values by creating random control variables (Big -Bang) phase. The Centre of Mass (CM) of Big-Bang phase is drawn into an ordered state by a Big- crunch phase.

Algorithm 1 BBBC Search Algorithm Workflow

1. Create random population of solution.
2. Evaluate Solutions.
3. The fittest individual can be selected as the centre of mass.
4. Calculate new candidates around the centre of mass by adding or subtracting a normal random number whose value decrease as the iteration elapse.
5. The algorithm continues until predefined stopping criteria has been met.

In fact, the Big Bang phase dissipates energy and produces disorder and randomness. In the Big Crunch phase, randomly distributed particles (which form the solution when represented in a problem) are arranged into an order by way of a convergence operator "centre of mass". The Big Bang–Big Crunch phases are followed alternatively until randomness within the search space during the Big Bang becomes smaller and smaller and finally leading to a solution. Above in the algorithm 2.1 is given the algorithm for the BBBC algorithm and the steps are included according to that particular algorithm.

III. Test Data Adequacy Criteria

Suitability of the individuals can be assessed by following a testing criterion for which a unique fitness function has to be defined. In structural testing, these criteria can be anything from all-statement-execution to all-path-coverage [Frankl1988]. We have chosen the all-path coverage criterion for our experimentation because one, it is the hardest to follow and second, in true sense, it is the real representative of structural testing. Very few test data generators have followed this criterion. The path testing method involves generation of test data for a target feasible path in such a way that on executing program, it covers all branches on that path. To cover a particular branch, the condition(s) at branch node must be satisfied by the test data, which directs the control flow of program to the next branch of the path. A path may contain several branches and in order to execute that path, all these branch- conditions must be evaluated true by the test data. Consequently, problem of path testing can be formulated simply as constraint satisfaction problem which should be analyzed and solved with the help of some search method by generating inputs in such a way that can satisfy all the branch constraints on the path.

A valid test case is generated, which should execute the particular path by satisfying all of the boolean expressions included in that path. Figure 1 shows the different building blocks of a path based automatic test data generator. First test object source code is fed to program instrumentation for CFG and node expressions generation. Subsequently CFG is used to generate all possible paths which are filtered manually for feasible path in order to become input to search algorithm. Node expressions include branch node predicates as well as non-branch node statements which are used to evaluate candidate solutions in test object fitness functions.

ISSN: 2321-9653

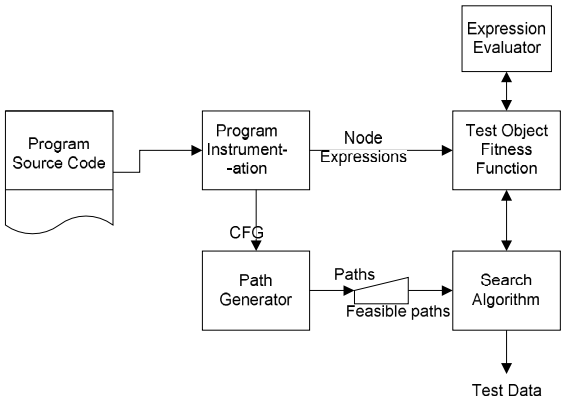


Figure 1 Path based Automatic Test Data generator

A. *Fitness Function*

For path testing criterion, in order to traverse a feasible path, the control must satisfy the entire branch predicates, which falls on that particular path. In our experimentations, we have used symbolic execution technique of static structural testing. So, corresponding to each path a compound predicate (CP) is made by '*anding'* each branch predicate of the path. The CP must be evaluated to true by a candidate solution in turn to become a valid test case. The BBBC generates population of candidate solutions and these are used to evaluate CP. If predicate is not evaluated to true by an individual then all the constraints of particular path are split into distinct predicate (DP) and one by one each DP is evaluated by taking values of its operands from candidate solution. A DP is that one, which contains only one operator (a constraint with modulus operator is exception) and can be expressed in form of expression A op B where A and B are LHS and RHS of expression made of one or more operand(s) and op is relational operator. If DP is satisfied then no penalty is imposed to candidate solution, otherwise candidate solution is penalized on the basis of branch distance concept rules as shown in table 1 which is also recommended by [Watkins et al 3] for static structural testing.

Table 1 Fitness Function For Branch Predicate

Violated Predicate	Penalty to be imposed in case predicate is not satisfied
$A < B$	$A - B + \zeta$
$A \leq B$	$A - B$
$A > B$	$B - A + \zeta$
$A \geq B$	$B - A$
$A = B$	$Abs(A - B)$
$A \neq B$	ζ- $abs(A - B)$

ζ is a smallest constant of operands' universal domain

After this integrated fitness due to whole of CP is determined by adding penalty values of two DPs, if they are connected by a conditional *'and'* operator. If two DPs are connected by a conditional *'or'* operator then minimum penalties of two DPs is considered for the evaluation of whole CP fitness. If integrated fitness is zero then CP is called evaluated otherwise search is allowed to proceed further.

IV. CONCLUSION

A new approach for automated test data generation is evaluated the name of which is Big Bang-Big Crunch algorithm is used for test case generation. Static testing based symbolic execution method has been used in which first, target path is selected from CFG of program and then inputs are generated using the BBBC method to satisfy composite predicate corresponding to the path. It has been observed that the BBBC method is better alternative than random testing.

REFERENCES

[1] J. Wegener, A. Baresel and H. Sthamer, "*Evolutionary test environment for automatic structural testing*," Information and Software Technology, 2001; Vol. 43.

[2] R. Pargas, M. Harrold and R. Peck, "*Test-data generation using genetic algorithms,"* Journal of Software Testing, Verification and Reliability 1999; 9(4): pp. 263–82.

[3] Surinder Singh and Parvin Kumar, "*Application of Big Bang-Big Crunch algorithm to software testing*", International journal of Computer Science and Communication, 2012; Vol. 3

[4] Hakki Murat GENC, Ibrahim Eksin and Osman Kaan Erol, *"Big Bang-Big Crunch optimization algorithm with local directional moves*", Turkish Journal of Electrical Engineering and Computer Science, 2013.

[5] Cheshta Jain, H.K. Verma and L.D. Arya, "*Big Bang-Big Crunch based optimized controller for automatic generation control and automatic voltage regulator system*", International Journal of Science and Technology, 2011; Vol.3

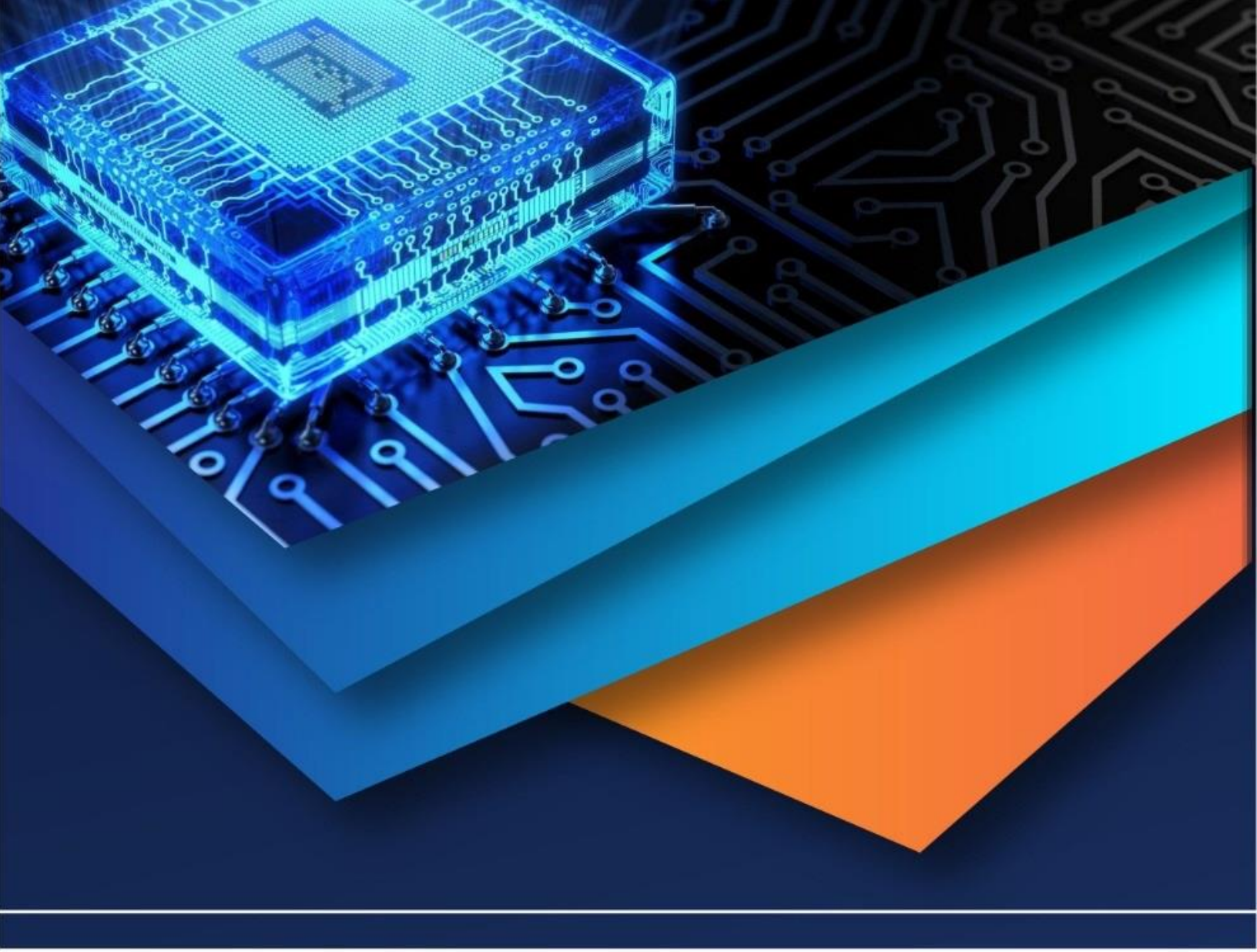
[6] K. Ayari, S. Bouktif and G. Antoniol, "*Automatic mutation test data generation via ant colony*", GECCO, 2011, London United Kingdom

[7] Abdelaziz M. Khamis, "*Automatic software test data generation for spanning sets coverage using genetic algorithm*", Computing and Informatics, 2007

[8] Pavel Y. Tabokov, *"Big Bang-Big Crunch optimization method in optimum design of complex composite laminates*", World Academy of science , engineering and technology, vol. 53, 2011

[9] S. Sakthivel, S. Arun Pandiyan, S. Marikani, S. Kurinji Selvi, "*Application of Big Bang-Big Crunch for optimal power flow problem",* International Journal of engineering and science, Vol. 2, page 41-44, 2013

[10] A. Kaveh, S. Talatahari, "*A discrete Big Bang-Big Crunch Algorithm for optimal design of skeletal structures*", Asian journal of civil engineering, Vol. 11, 2010



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY