A Survey on Automatic test Data Generation for MC/DC criteria using Search-Based Techniques

Nandita Priyadarshini¹, Mitrabinda Ray ²

¹Department of Computer Science Engineering, Siksha ‘O’ Anusandhan University (India)
²Department of Computer Science Engineering, Siksha ‘O’ Anusandhan University (India)

Abstract
The Search-Based Software Engineering (SBSE) techniques have been tested strongly on software engineering problems. It exercises on quality assessment, software preservation and requirement engineering. A field where the SBSE has seen enough operation is on generation of test data. An automatic generation of test data approach is applied to cut down the number of attempts for testing. The Modified Condition/Decision Coverage (MC/DC) criterion is one of the form of white box testing approach which is applied to present the coverage by testing whole conditions are associated in the predicate can affect the predicate value. MC/DC is a classic condition/decision coverage approach. One requirement of MC/DC is test cases should be elected to determine that a condition can independently influence to the result of a decision. Now days the meta-heuristic search approaches are applied for automatic test data generation. It has been a prospering activity for analysts or scientists. The Meta-heuristic search approaches are high-level schemes, which is applying heuristics i.e. multiple objectives to explore results for combinatorial problems at a feasible computational cost. The meta-heuristic search approaches are generally used for generation of test data.

Keywords: Search-Based Software Engineering (SBSE), Meta-Heuristic search techniques, MC/DC Criteria, Automatic Generation of the Test Data.

1. Introduction
The SBSE approaches have been used for different software engineering activities [31]. One main field where SBSE has seen many operation on generation of test data. It applies meta-heuristic search approaches like Hill Climbing (HC), Simulated Annealing (SA), Taboo Search (TS) and Genetic Algorithm (GA) for software engineering problems.

Many software engineering activities can be declared as optimization problems [22]. The operations of optimization approaches such as dynamic programming or linear programming are commonly illogical for a big scale software engineering problems because of their computational complexity. Researchers use meta-heuristic search techniques to search close-optimal solutions. SBSE problems can be divided into two types:

- White-box problems, where operations on source code need to be examined.
- Black-box problems, where the tests are placed on the functional needs of an operation.

The SBSE converts a software engineering problem into a computational search problem. It can be handled by meta-heuristic approaches. This associates with set of possible solutions or characterize a search space. This space is generally too big to be analyzed sincerely, implying a meta-heuristic approach. The SBSE technique is a redevelopment of software engineering process like search problems i.e. searched by examining a big search space of desirable results. According to Harman and Jones, meta-heuristic search approaches are applicable for many software engineering programs are associated with search and optimization problems because generally that is needed to balance challenging constraints and handle with inequality without any actual rules to figure out the best result [21]. Generally there are too many solutions without an ideal one, but still good results can be identified from the bad ones. The most used search techniques in SBSE are HC, GA, SA and TS etc. Different methods based on meta-heuristic techniques have been successfully applied on software engineering problems. One challenge of SBSE is to redevelopment of software engineering problems like search problems.

The research on Search-Based Software Testing (SBST) has engaged the researcher’s attention in current years as a type of a common significance in SBSE approaches [22]. The burgeoning significance of SBST can be applied on matters that there should be a commitment for automatic test data generation, so this is strongly called as extensive testing which is not feasible. Automatic generation of test data is treated as NP-hard problems [23].

Search Based technique is applied for Structural Based Testing; I’m going to introduce Structural Testing below. Also it is known as white box testing or glass box testing, where the test cases are derived from the ability of the software’s architecture or internal implementation. There is several other name of structural testing that includes logic-driven testing, open-box testing, and clear-box testing or path-driven testing. Types of Structural Testing approaches are:
• **Statement Coverage**: The main objective of this technique is to applying all the programming statements available in a code with minimum tests. It is the weakest form of testing because it needs that each statement in the code has been executed partially once.

• **Branch Coverage**: This approach is executing list of tests to ensure that whole branches are tested at least once. But it is not suitable for multiple condition checking.

• **Path Coverage**: This approach is responsible for testing all the desirable paths which means that whole statement and branch are covered. But this process is complex on when a loop is occurred.

• **Condition Coverage**: It checks all the possible conditions of a combination. In this approach, if the decision branch contains very complex Boolean expressions or many sub expressions, then the tester will define a large number of test cases. To reduce test cases, MC/DC criterion was constructed by J.J. Chilenski and S.P. Miller [19]. Only on time it changes the value of only one condition. So fewer test cases are needed to fulfill this criterion’s requirement. This technique is powerful and necessary for testing. With MCDC, each desirable result of a condition regulates partially once result of a decision. It signifies Condition/Decision Coverage. Especially it covers, in each condition when TRUE determines a TRUE result for entire decision, and when FALSE determines a FALSE result for entire decision. The same step goes for all other conditions in the decision point present in code.

2. **Background**

In SBSE, the word ‘search’ is applied to indicate the meta-heuristic Search-Based Optimization (SBO) approaches. SBSE explores to redevelopment of Software Engineering Problems as SBO problems of SBSE, a search problem is a type of problem in which excellent or closely excellent results are needed in a search space of candidate results, i.e. handled by a fitness function. This discriminates between the good and the worst results. [22] Search-Based software for generation of test data is an example of SBSE. Up to now, meta-heuristic approaches have been used for automatically generation of the test data in following areas:

- Attempting to contradict, undoubtedly certain grey-box properties are relating with application for a type of software.
- The applying of some particular program quality, as characterized by a specification.
- The coverage of particular program architecture, as part of a white-box or structural testing method.
- To certify non-functional properties.

2.1 **Meta-heuristic Search Technique**

Utilization of the meta-heuristic search approaches for automatically generation of the test data have been a prospering activity area for analysts or scientists in nowadays [31]. The operation of meta-heuristic search approaches for generation of test data is a probability which provides guarantee for the problems. Meta-heuristic search approaches are high-level scheme which is used heuristics, especially to search the results of combinatorial problems at a feasible computational cost. The meta-heuristic search approaches have been used to the testing problem in an area that is called SBST; it is a sub-group of SBSE. Evolutionary algorithm is the most suitable for meta-heuristic search approaches. It is extensively applied for explaining different kind of problems. Followings are different local Search approaches that are generally applied to solve the problems;

i. Simulated Annealing  
ii. Taboo Search  
iii. Hill Climbing  
iv. Genetic Algorithm

2.1.1 **Simulated Annealing (SA):**

It is a random search approach which feats a comparison between how a metal cools chills in to minimum number of energy crystalline structure and the analysis of a minimization of other system; it creates the base for an optimization process for combinatorial and other problems [34]. SA’s main feature is power to avoid becoming at bay in local minima [35]. It applies a random search technique which accepts the modifications that decrease the objective function f, assuming a minimization problem, but also some modifications increases the technique. The recent are accepted with probability \( p = \exp (-df / C) \) where \( df \) is the improvement in \( f \) and \( C \) is control parameter, in which by comparison with actual operation is called as system “temperature”, which is irrespective to the main function is engaged. The implementation key of SA algorithm is straightforward.

The following elements must be provided:

- A generator for random changes in results  
- A representation for possible results  
- Evaluation of the problem operations  
- An annealing agenda – a starting temperature and rules for letting down it as the search progresses.
2.1.2 Taboo Search (TC)

Taboo search is a continual process for explaining various combinatorial optimization problems. The space of whole desirable results is found in an array for changes from one probable result to the optimum feasible substitute. In order to avoid being battered at a sub-optimal result and to prevent floating off from the global best solution and some changes are divided as taboo or forbidden. It has been used in the work of for structural software testing [32].

2.1.3 Hill Climbing (HC)

It applies to enhance the result with a basic result that is randomly selected from the search space as an initial point [22]. It is known as local search algorithm. In this case search continues randomly, from a chosen place by considering the nearer point. Once a better neighbor point is searched, it converts in to present stage in the place of search space, and then the procedure is repeated. If there are no better neighbor points are present, then search concludes and maximum number of search spaces has been found. Still, it is not complicated which is very simple to use and prosperous in software engineering operations for modularization and calculation of the cost [31]. Fig.2 is the structure of Hill Climbing Algorithm;

2.1.4 Genetic Algorithm (GA)

GA is a heuristic search approach and also it is a form of an evolutionary algorithm. The main idea of the algorithm is that to begin with an arbitrarily computed number of individuals. Each individual has a possible result for a given problem [5]. A fitness function is applied to calculate the competency and aspect of each individual. The three main genetic functions are (i) selection, (ii) crossover and (iii) mutation to carry out the search. In selection process, fitness function is combined with each individual, to pull out a subgroup from a recent population. It means that better results are more expected to be chosen. These chosen individuals are mixed to create a new generation of population [12]. The combination is generally done through crossover functions, which takes two individuals and alternate their information at a probable chosen position. Generally a mutation process is used, to avoid that individuals become too similar so the population is calculated again, and the process is repeated until a specific completion condition is fulfilled. In Fig.3 the basic steps of GA process are shown below:

In the above figure, an initial population of individuals, each represented by a randomly generated individual, is created. The GA starts to evolve good solutions from this initial population. The three basic genetic operators: selection, crossover and mutation carry out the search. Selection of a string depends on its fitness relative to that of other strings in the population. The genetic search process is iterative: evaluating, selecting and recombining...
strings in the population during each generation until reaching some termination condition. Evaluation of each individual is based on a fitness function that is problem dependent. Selection is done on the basis of relative fitness. It probabilistically culls from the population individuals having relatively low fitness. Mutation flips a bit with very low probability and insures against permanent loss of alleles. One type of simple crossover is implemented by choosing a random point in a selected pair of strings and exchanging the substrings defined by that point. Then modify the population according to the fitness function.

2.2 Coverage Criteria
There are several coverage criteria to be considered in search-based software testing as follows:

2.2.1 Statement Coverage
In a program every statement has been executed partially once. When each statement of a code is executed, 100% statement coverage is achieved.

2.2.2 Decision Coverage
In a program every statement has been executed partially once, and each decision of a program has taken whole desirable results partially once.

2.2.3 Condition Coverage
In a program every statement has been executed partially once, and every condition in each decision has taken whole desirable results partially once.

2.2.4 Decision/Condition Coverage
In a program every statement has been executed partially once, each decision of a program has taken whole desirable results partially once, and every condition in each decision has taken whole desirable results partially once [19].

2.2.5 Modified Condition/Decision Coverage
In a program every point of start and end have been appealed partially once, each condition in a decision has taken whole desirable results partially once, each decision has taken whole probable results partially once, and each condition in a decision has been shown independently to influence the results of a decision [19]. To the best of my knowledge Search-Based Techniques are not widely used for MC/DC criteria. The detailed coverage criteria for decision making are shown in table 1.

Table 1. Various coverage criterions applied on search-based software testing (SBST)

<table>
<thead>
<tr>
<th>Coverage Criteria</th>
<th>Statement Coverage</th>
<th>Decision Coverage</th>
<th>Condition Coverage</th>
<th>Condition/Decision Coverage</th>
<th>MC/DC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Every point of entry if</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>and exit in the program</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>has been invoked at least</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>once</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Every statement in the</td>
<td></td>
<td>*</td>
<td></td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>program has been</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>invoked at least once</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Every decision in the</td>
<td></td>
<td>*</td>
<td></td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>program has taken all</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>possible outcomes at least</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>once</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Every condition in the</td>
<td></td>
<td>*</td>
<td></td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>decision has taken</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>all possible</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>outcomes at least once</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Every condition in a</td>
<td></td>
<td>*</td>
<td></td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>decision has been shown</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>to independently affect</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>decision’s outcome</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The critical software is badly requiring the Condition Coverage to operate them correctly. In Condition Coverage, if the decision branch contains very complex Boolean expressions or sub-expressions, then the tester will have to define large test cases. To reduce large test cases, the NASA members were proposed MC/DC [20]. It is a powerful and an important criterion to checks the conditions and decisions depending on a variable. Only one time it changes the value of only one condition’s. So it needs less number of test cases to qualify this criterion, this technique is powerful and necessary for testing.

3. Test Data Generation using Search-Based Testing Techniques

Now days, different works are done using meta-heuristic algorithms as search tools to automatic generation of test data. According to the computational complexity for the search problem, the specific approaches such as linear programming are mostly illogical for a big scale of software engineering problems and manual search technique is not possible. So SBSE is a technique to use the meta-heuristic search approaches such as GA, SA, HC and TS to extract the results for combinatorial problem to a desirable computational cost. In SBSE, search techniques are used to find large search spaces, handled by a fitness function that analyze the results with respect to the search objective and demonstrate which is better results for the problems. So the automatic search is a possibly encouraging field for search space.

For generation of test data, it involves the conversion of test criteria to main operations. For each test criterion, a group of different main function is required. Then the algorithm emphasizes to generate an appropriate data to create the fitness function as near as possible to zero. Each problem is solved using meta-heuristic approaches; there are generally two main decisions of implementation to be used. The first one is encoding of the result, i.e. the structure like tree, array, how many variables it has, their types etc, and the second main decision is the conversion of the test criteria into a fitness function. It is generally
evaluated at the end of the each algorithm and it analyzes and compares the results with respect to whole search goal to guide the search into an encouraging neighborhood of the search space. There are different types of metaheuristic algorithms that were applied for automatic generation of test data.

3.1 Application of Genetic Algorithm for Testing on MC/DC Criteria

Most of the time for structural testing the branch coverage criterion is used. Still, an individual criterion is insufficient for an efficient testing. For broadly approval of the Search-Based Test Data Generation approaches, a very stronger criterion is required such as MC/DC. It needs few test cases to fulfill this criterion. This technique is very powerful and necessary for testing. Only one time it changes the value of only one condition’s. The Fig.4 shows the process involved in genetic algorithm satisfying MC/DC criterion.

![Figure 4](Image)

Figure 4 steps involved in genetic algorithm satisfying modified condition decision coverage.

In the above figure, one existing code is taken for the testing. After that test cases are generated using genetic algorithm, because it generates number of optimal solution for a problem. The GA starts to evolve good solutions from this initial population. The three basic genetic operators: selection, crossover and mutation carry out the search.

- **Test-case evaluation**: The resulting set of test cases produced by the GA needs to be executed in order to compute their fitness and also to check the correctness of the results.

- **Test-coverage computation**: The analysis of the program under test is performed manually for the time being, and all the information is stored in files for later use by the GA. Although this may sound very time-consuming, it does not actually affect the execution time of the GA, since this process takes place only once and prior to the execution of the algorithm.

### 4. Comparison Study on the Existing Test Data Generation Based On Search Based Techniques:

<table>
<thead>
<tr>
<th>Work</th>
<th>Objective</th>
<th>Search Techniques</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1]</td>
<td>Using Multiple Conditions and MC/DC Criteria for automatic test data generation</td>
<td>Search approach applied SA. Proposed framework [tool]</td>
<td>The author has developed a flexible prototype tool. That tool has determined how the search-based test data generation approach can be applied on generation of test data at the lowest level of the branch.</td>
</tr>
<tr>
<td>[3]</td>
<td>Comparison of two Fitness Functions for GA-based Path-oriented Test Data Generation</td>
<td>Genetic Algorithm, Normalized Extended Hamming Distance and Branch Distance</td>
<td>It works on big and more complex programs.</td>
</tr>
<tr>
<td>[5]</td>
<td>An approach to Generate</td>
<td>Genetic Algorithm</td>
<td>To decrease evolutionary status and to enhance the...</td>
</tr>
<tr>
<td>Reference</td>
<td>Description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[6]</td>
<td>Automating test data generation for C programs using BLAST (software model checker) to develop a tool in order to fulfill the requirement of criteria, including boundary value analysis, equivalence partitioning, defuse analysis, and search for probable errors in code.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[8]</td>
<td>Optimizing the number of tests generated in search-based test data generation with an application to the Oracle cost problem using a hybrid algorithm that merges the advantages of both CDG and set cover techniques.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[9]</td>
<td>Breeding software test data with genetic particle swarm mixed algorithm (GPSMA) for increased MC/DC coverage using concolic testing, CREST tool, and coverage analyzer.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[10]</td>
<td>An algorithm for heuristic approaches to improve the accuracy and scalability of the testing process.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[15]</td>
<td>Test data generation for Automatic software structural testing using Evolutionary algorithm</td>
<td>When generation of new population is placed on choosing those individuals with high fitness, this gives the opportunity to generate a better offspring. Genetic Algorithm executes better offspring from one generation to the next generation to cover chosen paths with difficult paths.</td>
<td></td>
</tr>
<tr>
<td>[16]</td>
<td>Using firefly algorithm for optimal test sequence generation</td>
<td>The author has extended the standard firefly algorithm to generate optimal discrete and independent paths for software testing. Graph reduction also helps to ensure the right code coverage for testing.</td>
<td></td>
</tr>
<tr>
<td>[17]</td>
<td>AUTOMATIC TEST DATA GENERATION USING CUCKOO SEARCH AND TABOO SEARCH (CSTS) ALGORITHM</td>
<td>Cuckoo Search uses cuckoo algorithm for selection and generation of candidate solutions guided by CDG to obtain path coverage criterion. Diversification in the search is achieved using backtrack operation with the help of taboo lists which explores new horizons of the search space and Levy flight is used to prevent the solution to being stuck in local optima.</td>
<td></td>
</tr>
<tr>
<td>[18]</td>
<td>Automatic test data generation for Unit</td>
<td>The coverage of the test data generation by MC/DC approaches are generally increases than the existing approaches.</td>
<td></td>
</tr>
<tr>
<td>[19]</td>
<td>Applicability of MC/DC criteria to software testing</td>
<td>The MC/DC criterion fills an important gap by providing better coverage of the underlying code than Condition/decision testing.</td>
<td></td>
</tr>
<tr>
<td>[20]</td>
<td>MC/DC for automatic test input data generation</td>
<td>The author has presented a unique fitness function by integrating data to generate test data for MC/DC criteria.</td>
<td></td>
</tr>
</tbody>
</table>

5. Summary
This paper has given an analysis of the SBSE and the SBST applied in generation of test data. One major objective is to do a review of applying SBO approaches to automatic modification of the results for Software Engineering Problems. Automatically search the test data, the Search-Based Test Data Generation investigates test data for input domain of the object. This process is managed by the fitness function. In this paper the focal point is GA for generation of test data. It is broadly applied for Search-Based approaches to the test data generation problems. GA maintains a number of results rather that just one recent result. So, the investigation is allowed from different starting points and an opportunity to sample more search space than local searches.

References


[8]. Harman, Mark "Optimizing for the number of tests generated in search based test data generation with an application to the oracle cost problem", Third International Conference on Software Testing, Verification, and Validation Workshops (ICSTW), IEEE, 2010.


