2011 International Conference on Software and Computer Applications
IPCSIT vol.9 (2011) © (2011) IACSIT Press, Singapore

Automated Test Data Generation and Optimization Scheme Using Genetic Algorithm

Roshni Rajkumari [1] and Dr.B.G.Geetha[2]

[1] M.E 2nd Yr, KSR College of Technology, Tiruchengode, 637215

[2] Professor,KSR College of Technology,Tiruchengode,637215

Abstract. Software testing requires test cases and test data values for functional and non-functional tests. Test automation schemes are used to generate test cases and test data automatically. Structural test data generation schemes use the program as input and produces test data collection as output. Optimal data selection is required for the test data generation applications. A variety of machine learning methods are used to filter optimal data values from the generated test data values. Search based test data optimization schemes are used to select optimal data values from the test data collection. Global search is used select global optima for the structural test. The local search is used to select local optima for the sub branches. The hybrid global local search scheme is used to select optimal data value for the structural analysis. The memetic algorithm is used for test data optimization.

Keywords: Automated test data generation, search-based, Genetic Algorithms, Hill Climbing, schema theory, Evolutionary Testing.

1. Introduction

There is strong evidence that deficient testing of both functional and nonfunctional properties is one of the major sources of software and system errors. One technique which has received much attention is that of applying search-based optimization to the problem of software test data generation. This process is known as Search-Based Testing. Search-Based Testing is the process of automatically generating test data according to a test adequacy criterion also known as a fitness function. The fitness function is to captures a test objective that, when achieved, makes a contribution to the desired test adequacy criterion. With this fitness function, the search seeks test inputs that maximize the achievement of this test objective.

Evolutionary Testing is a subfield of Search-Based Testing. It has been applied to many test data generation scenarios including temporal testing, stress testing, and exception testing. Since Evolutionary Testing uses a global search approach to find structural test data, the efficiency and effectiveness of Evolutionary Testing is to be compared with local search technique. The most widely studied local search technique for search-based test data generation is Korel's Alternating Variable Method. This is an approach which is a form of Hill Climbing. Global search technique overcome the problem of local optimum in the search space and can find more globally optimal solutions. Local search may become trapped in local optima within the solution space.

Global search involves a collection (a "population") of candidate solutions that evolve over time, allowing for a wide sampling of the search space. By contrast, local search uses the fitness function to evaluate possible moves within the search space from a single current solution point until a local optimal is reached. To solve this, the paper presents a theoretical development of Holland's schema theory. The theory was later developed by Mitchell. The crossover operator attempts to build fitter (i.e., better) candidate solutions from good solutions present in the current population. It does so by recombining the elements that make up the "chromosome" of each solution. Both the schema theory and the Royal Road theory were

developed for chromosomes represented as bit strings, and have not been previously adapted for the more complex chromosomes required by Evolutionary Testing.

The paper presents plays two roles: It validates the predictions of the theory and it answers the questions concerning the relative performance of global and local search. Local search can be very effective and efficient, but there remain Search- Based Testing optimization problems for which global search is the only technique that can successfully achieve coverage.

2. Search-Based Testing

Search-based test data generation searches a test object's input domain to find test data automatically. It is guided by a fitness function. This paper concentrates on structural test data generation. It is most widely studied of all the applications of search-based techniques to the test data generation problem. It considers branch coverage, a widely used structural test adequacy criterion. The results can also be extended to apply to other forms of structural test data generation.

2.1. Genetic Algorithm and Evolutionary Testing

Genetic Algorithms is an optimization technique that shows the process of natural evolution. It is used to generate useful solutions to optimization and search problems. The search simultaneously evolves several individuals in a population, creating a global search. This section begins with a description of a basic Genetic Algorithm and then explains how the Genetic Algorithm used by Evolutionary Testing differs. In the Genetic Algorithm the first stage is the initialization of a population of n candidate solutions, known as "individuals," at random. The chromosomes representing each individual are encoded as bit strings for manipulation by the algorithm. After initialization, the Genetic Algorithm enters a loop, comprising of distinct stages of evaluation, selection, crossover, mutation, and reinsertion.

In evaluation phase, each individual is assessed for fitness using the fitness function. Selection is the process of choosing "parent" candidate solutions which will be "bred" in the crossover phase to produce offspring solutions, and then, subject to a stage of mutation. Crossover loosely models the exchange of genetic information that takes place during reproduction in the natural world. There are many choices of crossover operator. Simple one-point crossover involves selecting a crossover point where two parent chromosomes are to be spliced in order to form the composite chromosomes of two children. Traditionally, this involves flipping bits in each individual's chromosome at a probability of pm, where pm is typically 1/len, where len is the length of the chromosomal bit string.

Crossover will simply combine the best features of both parents to create super-fit children from fit parents, and that mutation will also help discover fitter individuals. Where this fails to take place, bias involved in the selection phase ensures that less-fit individuals have less chance of being selected to reproduce in the next iteration of the algorithm and thus "die out". One type of selection process is fitness-proportionate selection. In this, the individuals are selected at a probability that is proportionate to their fitness value compared to other individuals in the population. But, overselection of the best candidate solutions may result in premature convergence if the populations become dominated by a few superfit individuals. Thus, ranking selection methods are often preferred. Individuals are ranked in fitness order, and are then chosen randomly at a probability proportionate to their rank. Reinsertion involves forming a new generation of individuals from the current population and the generated offspring. One approach is to replace all of the current population with offspring; another approach is where the best individuals are retained, taking the place of some of the weakest newly generated offspring, which are discarded.

Finally, the new individuals of the population are evaluated for fitness. At each evaluation stage, a test is performed to see if the goal of the search has been met, i.e., the global optimum has been found, or if the search has failed, and should be terminated. Termination conditions tend to test if a certain number of trials (fitness evaluations) have been performed, or a certain number of loops of the algorithm ("generations") have been performed.

Evolutionary Testing is the application of Genetic Algorithms to Search-Based Testing. It is also known as Evolutionary Testing. Evolutionary algorithms can assist in finding test cases which cover the code base

under test to a maximum extent . The aim is to execute the code under test with as many different input parameters as possible, in order to maximize the chance of detecting errors in the code.

Evolutionary algorithms use the principles of evolution to perform optimization based on the result of a fitness function. The fitness of a first generation of random individuals is tested, and the characteristics, known as genes, of the fittest individuals are propagated to the next generation. This process is governed by rules regarding which percentage of individuals have their genes propagated to the next generation, how their genes are combined to form the next generation's individuals and how the genes are randomly mutated.

For Evolutionary Testing, the chromosome making up each individual is a direct representation of the input vector to the program concerned. The "genes" of the chromosome represent the input values with which the program will be executed because test data generation requires chromosomes that must respect typing information embodied in any valid input type.

The evolutionary algorithm completes when one of several criteria has been met, e.g. individuals have been found which cover all outcomes from the branch statement under test, or not all outcomes have been covered and no improvement in the fitness of the individuals has occurred in the last generations. The search is repeated for each branch statement of the function. Finally the group of test cases covering as many branches of the function as possible is presented to the user.

2.2. Hill Climbing

Hill climbing is a mathematical optimization technique. It belongs to the family of local search. It is an iterative algorithm that starts with an arbitrary solution to a problem that attempts to find a better solution by incrementally changing a single element of the solution. If the change produces a better solution, an incremental change is made to the new solution, repeating until no further improvements can be found.

Hill Climbing improves a single-candidate solution, starting from a randomly selected starting point. From the current position, the neighboring search space is evaluated. If a fitter candidate solution is found, the search moves to that point. If no better solution is found in the neighborhood, the algorithm terminates. The method has been called "Hill Climbing" because the process is like climbing of hills on the surface of the fitness function (referred to as the "fitness landscape").

One problem with Hill Climbing is that they become trapped in a local minima. Hill climbing attempts to maximize (or minimize) a target function $f(\mathbf{x})$, where $\mathbf{x}$ is a vector of continuous and/or discrete values. At each iteration, hill climbing adjust a single element in $\mathbf{x}$ and determine whether the change improves the value of $f(\mathbf{x})$. With hill climbing, any change that improves $f(\mathbf{x})$ is accepted, and the process continues until no change can be found to improve the value of $f(\mathbf{x})$. $\mathbf{x}$ is then said to be "locally optimal".

In order to solve this problem, Hill Climbing is restarted at a new randomly chosen start point many times, until no changes is found.

2.3. Hybrid Memetic Algorithm Approach

Memetic Algorithms are Evolutionary Algorithms which uses local search to improve each individual at the end of each generation. The Memetic Algorithm used combines the Evolutionary Testing and Hill Climbing methods. However, there are some modifications in order to balance the new hybrid algorithm's abilities to intensify the search, i.e., to concentrate on an explicit subregion of the search space, and diversify the search, i.e., explore new and unseen areas.

Some problem with the Hill Climbing is that it terminates for each individual upon reaching local optima and does not restart and a smaller population size of 20 is employed, without the use of subpopulations. In the hybrid algorithm, Hill Climbing is used to intensify the search on particular areas of the search space. Since diversification does not happen until the end of each generation, a reduced population size is necessary to prevent the search spending the majority of its time intensifying around the space of its current set of individuals. Finally, the Breeder Genetic Algorithm mutation operator is replaced with uniform mutation, which is used for greater diversification, balancing the high intensification of the Hill Climbing phase. Uniform mutation simply involves overwriting an input variable value with a new value from its domain, chosen uniformly at random.

3. Theoritical Foundation

This section presents an overview of the schema and Royal Road theories and introduces a generalization of both that caters for Evolutionary Testing.

3.1. The Schema Theory of Genetic Algorithm

In a binary Genetic Algorithm, a schema is a sequence consisting of three possible values, drawn from the set {0, 1, *}. The asterisk is a wildcard, indicating that either a zero or a one could occur at this position. Thus, for a chromosome of length 4, a schema 10 * 1 denotes the two chromosomes 1011 and 1001, while the schema **** denotes all possible chromosomes. A schema can be thought of as a template chromosome that stands for a whole set of individual chromosomes, each of which share some common fixed values. An instantiation of a schema is any chromosome that matches the template, when * values are replaced by the corresponding fixed values of the instantiation.

If a chromosome x is an instantiation of a schema h, this is denoted as $x \in h$. The number of fixed positions in a schema is called the order of the schema. The schema 1**1 has order two, while 10*1 has order three. For a schema h, the order will be denoted by o(h). Suppose that the length of a chromosome is denoted by λ . A schema h denotes $2^{\lambda - o(h)}$ chromosome instantiations, all of which have their own individual fitness values.

3.2. Schema Theory for Test Data Generation by Genetic Algorithm

Evolutionary Testing does not use binary Genetic Algorithms, so the schema theory is not directly applicable. A new form of schema theory for Evolutionary Testing therefore has to be constructed. Fortunately, the input vectors used in Evolutionary Testing can be captured by a generalization of Holland's schema theory. These Evolutionary Testing Schemata arise from the constraints on the input that a branch-covering solution must satisfy. The constraints can be defined naturally in terms of the computation of fitness for the approach level and branch distance computation. For example, suppose a program has three inputs, x, y, and z and that, in order to execute the branch under test B, the program must first follow a branch B_1, for which the condition x > y must hold, and must then follow a branch B_2, for which the condition y = z must hold.

3.3. Royal Road for Evolutionary Test Data Generation

For a schema concerned with constraints, the order of the schema is the number of variables that are mentioned in the constraint. In order for lower order schemata to be combined with higher order schemata, as with the binary Genetic Algorithm model, the higher order schemata must contain the union of the genes of the lower order schemata. Also, to avoid destroying the properties of a lower order schema, there must be no intersection of genes in the lower order schemata; otherwise, the genes of one would overwrite those of the other when combined. This is also the case with the Royal Road theory.

For an Evolutionary Testing approach to exhibit a Royal Road property, the more fit schemata must be expressed as conjunctions of lower order schemata. Where this property holds, the Evolutionary Testing Royal Road theory predicts that Evolutionary Testing will perform well and it will do so because of the presence of the crossover operation and the way in which fitter schemata are given exponentially more trials than less fit schema.

4. Proposed Test Data Generation And Optimization Scheme

Software testing requires test cases and test data values for functional and non-functional tests. Test automation schemes are used to generate test cases and test data automatically. Structural test data generation schemes use the program as input and produces test data collection as output. Optimal data selection is required for the test data generation applications. A variety of machine learning methods are used to filter optimal data values from the generated test data values. Search based test data optimization schemes are used to select optimal data values from the test data collection. Global search is used select global optima for the structural test. The local search is used to select local optima for the sub branches. The hybrid global local search scheme is used to select optimal data value for the structural analysis. The memetic algorithm is used for test data optimization. The fitness functions are used to select suitable data values.

The proposed system is designed to perform test data generation and optimal data selection process. The pseudo Oracle transformation is used to identify the global data elements. The optimal data selection process is enhanced to select global data elements and branch based data elements. The genetic algorithm is enhanced with mixed mutation scheme.

Structural functional testing is carried out in the system. Test data values are generated from the branch information. Test cases are automatically identified by the system. Genetic algorithm is used to perform optimal data selection process. The system is divided in four modules:

- Program analysis
- Test constraints Extraction
- Test data generation
- Optimal data selection

Source code analysis fetches the branch information. Java source code is used as the input for the system. Branching statements and looping statements are used to build branch hierarchy. Program identifier information is also analyzed. Test cases are identified using the branching information. Pseudo oracle transformation is carried out to identify the global data elements. Constraints in each code branch are assigned in the test constraint list. Test schema is build with the constraint information. Nested branch information is also analyzed for the test constraint identification process.Test schema is used for the test data generation. Random data values are assigned with the schema. Test constraints are also considered in the test data generation process. Test data values are updated in the data list repository. The genetic algorithm is used to select optimal data values. Chromosomes are build with test data information. Crossover and mutation operations are performed in each generation. Local and global search models are used for the optimal data selection process.

5. References

[1] The Software-Artifact Infrastructure Repository, http://sir.unl.edu/portal/index.html, 2009.

[2] J. Aguilar-Ruiz, I. Ramos, J.C. Riquelme, and M. Toro.An Evolutionary Approach to Estimating Software Development Projects *Information and Software Technology*. vol. 43, no. 14, pp. 875-882, 2001.

[3] G. Antoniol, M.D. Penta, and M. Harman,.Search-Based Techniques Applied to Optimization of Project Planning for a Massive Maintenance Project. *Proc. IEEE Int'l Conf. Software Maintenance*, pp. 240-249, 2005.

[4] M. Harman ,The Current State and Future of Search Based Software Engineering. *Proc. Int'l Conf. Future of Software Eng*., 2007.

[5] J.E. Baker, "Reducing Bias and Inefficiency in the Selection Algorithm," *Proc. Second Int'l Conf. Genetic Algorithms and Their Applicatio*n, 1987.

[6] A. Baresel and H. Sthamer,.Evolutionary Testing of Flag Conditions. *Proc. Genetic and Evolutionary Computation Conf*., pp. 2442-2454, 2003.

[7] L. Bottaci,.Instrumenting Programs with Flag Variables for Test Data Search by Genetic Algorithm. *Proc. Genetic and Evolutionary Computation Conf*., pp. 1337-1342, 2002.

[8] S. Bouktif, G. Antoniol, E. Merlo, and M. Neteler,.A Novel Approach to Optimize Clone Refactoring Activity. *Proc. Genetic and Evolutionary Computation Conf*., pp. 1885-1892, 2006.

[9] S. Bouktif, H. Sahraoui, and G. Antoniol,.Simulated Annealing for Improving Software Quality Prediction. *Proc. Genetic and Evolutionary Computation Conf*., pp. 1893-1900, 2006.

[10] R. Ferguson and B. Korel.The Chaining Approach for Software Test Data Generation,.*ACM Trans. Software Eng. and Methodology*, vol. 5, no. 1, pp. 63-86, 1996.

[11] A. Finkelstein, M. Harman, A. Mansouri, J. Ren, and Y. Zhang,.Fairness Analysis in Requirements Assignments. *Proc. IEEE Int'l Requirements Eng. Conf*., 2008.

[12] R.L. Glass, Facts and Fallacies of Software Engineering. Addison Wesley, 2002.

[13] D. Greer and G. Ruhe,.Software Release Planning: An Evolutionary and Iterative Approach.,*Information and Software Technology*, vol. 46, no. 4, pp. 243-253, 2004.

[14] M. Harman.The Current State and Future of Search Based Software Engineering.*Proc. Int'l Conf. Future of Software Eng*. 2007, pp. 342-357, 2007.