Reduction and modification of Test Cases in Web Applications by Using Multi Objective Genetic Algorithm

Alireza Souri [1], Mohammad esmaeel Akbari [2], Arash Salehpour [3]

[1,3] Department of Computer Engineering, Ahar Branch, Islamic Azad University, Ahar, Iran
[2]Department of Electrical Engineering, Ahar Branch, Islamic Azad University, Ahar, Iran
E-mail: a-souri@iau-Ahar.ac.ir, m-Akbari@iau-Ahar.ac.ir, a-salehpour@iau-Ahar.ac.ir

Abstract: Web applications have countless constraints. Cost, time, and space constraints restrict us to execute all the test cases. We need to reduce the test cases to an appropriate amount so the efficient testing of web application can be done. Test case reduction is the process to extract the valid solutions eliminating redundancy and invalid solutions. This paper presents an idea to induce the intelligent aspect in automated testing by applying Multi Objective Genetic algorithm (MOGA). The term Multi objective here suggests multiple tasks to be achieved with efficiency. The multi-objective factors being considered here are cost and coverage. The cost at the end will be reduced cause of the prioritized test cases and coverage will be maximized as we will select the test cases with highest fitness. As per Genetic algorithm the initial population is the test cases extracted from web application. Fitness criterion applied is made on the basis of test case length i.e. the number of ids in each test case and then the test cases with the more length are selected for genetic operations. Pairs for cross over are made on the basis of their affinity. Mutation is applied for the diversity or to extract the right solution. It is not mandatory if required solutions are obtained. The system made for reduction of test cases is efficiently giving the accurate results and with maximum objectives being achieved.

[Alireza Souri, Mohammad esmaeel Akbari. **Reduction and modification of Test Cases in Web Applications by Using Multi Objective Genetic Algorithm.** Journal of American Science. 2012; 8(4):757-762]. (ISSN: 1545-1003). http://www.americanscience.org. 100

Keywords: Test suite prioritization, Multi objective genetic algorithm, Web applications, test case

1. Introduction

Web applications have distributed and heterogeneous properties which make them more interactive with users and also dominant than traditional ones. [1] Web applications are now responsible for performing the crucial tasks and it is not reliable until or unless secure systems are ensured. Apart from security other quality attributes like functionality are also necessary for the efficient systems. The field of web development is expanding day by day and in parallel to this the need of testing is also being increased. “Software Testing is the process of executing a program or system with the intent of finding errors.”[2].

Software testing is the process of finding the defects and faults. By removing these defects we can avoid maximum risks possible. Fault free implementation with the required functionality and specification is the foremost demand of the customers and this is only possible when you apply best testing techniques. Before the modern techniques of testing it was only applied at the later stages and was considered to be as a single stage of software development life cycle (SDLC) but now testing is applied at each phase for the required output. Finding out the errors in the initial stages saves 80% of time and cost as compared to the errors found in the end. Considering the test adequacy criteria’s for the testing, it can provide coverage through statement decision or paths. It follows the hierarchy on the top is multiple condition coverage and at the bottom is functions i.e. functions are not being repeated. Path testing is considered effective of all but that is bit impossible. Testing of decisions might be more complex than others. Before Automated testing, it was done manually. Manual testing is a lengthy and complex method. 60-40 percent of the project time is consumed by this process. A simple program may have many different behaviors and testing that program can turn out to be very expensive and time consuming.

If we consider testing huge software by manual testing it will be very exhausting and can be never ending task cause in such cases you are not sure when to stop testing. Human testing has more chances that some errors remain unidentified and it takes twice the resources as compared to automated testing. To take the work load off your hands and to do it in more efficient way we use automated testing. Automated testing is the process of record and run. It lowers the cost and time and provides more of the test coverage. It tests the program we feed in it. We give the proper input and look for expected output. Once the test case is give human work is over, through automated testing all the processing will be done.

User will just have to check the results in the by examining the successful and failed test cases.[3]

Automated testing has become vast field and much work has been done until now. Different web application testing software with modified versions are being released every next day.

Automated web application testing with respect to artificial intelligence is still an emerging field and not many liable and authentic testing frameworks have been made yet. This paper presents an AI based testing technique for the testing of web applications. AI techniques evolved from the natural phenomena's are most renowned and reliable in the field of automated testing. Genetic algorithm, Simulated annealing, PSO etc are some of the examples. The technique used in this paper is Multi Objective Genetic Algorithm (MOGA). This technique is used for the reduction of test cases .

Reduction of test cases means to trim down the test cases i.e. eliminate invalid and repeating test cases and also to extract the favorable valid test cases from the valid ones. Reduction has the benefit that it saves many resources as in time cost and effort and to save maximum of these resources we have made our algorithm multi objective.

After a decade since the pioneering work by Schaffer (1984), a number of studies on multiobjective genetic algorithms (GAs) have emerged. Most of these studies were motivated by a suggestion of a non-dominated GA outlined in Goldberg (1989). The primary reason for these studies is a unique feature of GAs—a population approach—that is highly suitable for use in multi-objective optimization. Since GAs work with a population of solutions, multiple Pareto-optimal solutions can be found in a GA population in a single simulation run. During the years 1993-95, a number of independent GA implementations (Fonseca and Fleming, 1993; Horn et al., 1994; Srinivas and Deb, 1995)[9] emerged. Later, other researchers successfully used these implementations in variousmulti-objective optimization applications (Cunha et al., 1997; Eheart et al., 1993; Mitra et al., 1998; Parks and Miller, 1998; Weile et al., 1996). A number of studies have also concentrated on developing new GA implementations (Kursawe, 1990; Laumanns et al., 1998; Zitzler and Thiele, 1998). Fonseca and Fleming (1995)[10] and Horn (1997) presented overviews of different multiobjective GA implementations, and Van Veldhuizen and Lamont (1998) made a survey of test problems that exist in the literature.

2. Related Works

Xu et.al.[1] Testing process for web application is explained in the given paper and has the following phases. Testing requirements i.e. developer should be well aware of its goal, method being used and object. After this, test cases are generated based on information provided by the previous phase combined with web application. These test cases are then reduced to relevant ones; selected test cases are then given the input information for the required output keeping the track of test steps. Then these cases are executed and monitored and the results deduced are then analyzed. If the results are like the original ones as expected, it is considered to be correct. Feedback is given after that.

Srivastava et.al. in [4], The main idea discussed in this paper is to find the most critical path and make test cases according to that. Critical suggests such paths which have most probability of errors. The approach used in this paper is weighted CFG. Critical paths are here found by assigning 80-20 rule. Genetic algorithm is then applied. In the process of selection fitness value is calculated by adding weights of each path. Random values are then generated and compared with cumulative probability Ci. Random number which is just lowest to the corresponding Ci that test data number is assigned to Ns column. Mating pool is the parent pool.

Test cases in the mating pool are crossover with the input test data cases if their random number is less than 0.8. For mutation each bit having random number less than 0.3 is flipped for new data entry. This technique can be used to produce optimal results also saving us from exhaustive testing by finding the critical paths in advance.

C. Michael et.al in [5], This paper uses the combinatorial optimization techniques such as genetic algorithm for the generation and prioritization of test cases. To start the implementation test adequacy criteria must be satisfied. Test adequacy criteria can be any coverage, criteria set in this paper is decision condition coverage. The test data generation method used is dynamic; it finds the desired location and applies function minimization. Genetic algorithms applied with function minimization follows its iterative process of initialization, fitness function, selection and genetic operators. All the results are then evaluated giving a clear margin that genetic search is more effective than random test data generation.

3. Genetic algorithms

The concept of GA was developed by Holland and his colleagues in the 1960s and 1970s [6]. GA are inspired by the evolutionist theory explaining the origin of species. In nature, weak and unfit species within their environment are faced with extinction by natural selection. The strong ones have greater opportunity to pass their genes to future generations via reproduction. In the long run, species carrying the correct combination in their genes become dominant in their population. Sometimes, during the slow process of evolution, random changes may occur in genes. If these changes provide additional advantages in the challenge for survival, new species evolve from the old ones. Unsuccessful changes are eliminated by natural selection.

In GA terminology, a solution vector xAX is called an individual or a chromosome. Chromosomes are made of discrete units called genes. Each gene controls one or more features of the chromosome. In the original implementation of GA by Holland, genes are assumed to be binary digits. In later implementations, more varied gene types have been introduced. Normally, a chromosome corresponds to a unique solution x in the solution space. This requires a mapping mechanism between the solution space and the chromosomes. This mapping is called an encoding. In fact, GA work on the encoding of a problem, not on the problem itself. GA operate with a collection of chromosomes, called a population. The population is normally randomly initialized. As the search evolves, the population includes fitter A. Konak et al. / Reliability Engineering and System Safety 91 (2006) 992–1007 993 and fitter solutions, and eventually it converges, meaning that it is dominated by a single solution. Holland also presented a proof of convergence (the schema theorem) to the global optimum where chromosomes are binary vectors. GA use two operators to generate new solutions from existing ones: crossover and mutation. The crossover operator is the most important operator of GA. In crossover, generally two chromosomes, called parents, are combined together to form new chromosomes, called offspring. The parents are selected among existing chromosomes in the population with preference towards fitness so that offspring is expected to inherit good genes which make the parents fitter. By iteratively applying the crossover operator, genes of good chromosomes are expected to appear more frequently in the population, eventually leading to convergence to an overall good solution. The mutation operator introduces random changes into characteristics of chromosomes. Mutation is generally applied at the gene level. In typical GA implementations, the mutation rate (probability of changing the properties of a gene) is very small and depends on the length of the chromosome. Therefore, the new chromosome produced by mutation will not be very different from the original one. Mutation plays a critical role in GA. As discussed earlier, crossover leads the population to converge by making the chromosomes in the population alike. Mutation reintroduces genetic diversity back into population and assists the search escape from local optima.

Reproduction involves selection of chromosomes for the next generation. In the most general case, the fitness of an individual determines the probability of its survival for the next generation. There are different selection procedures in GA depending on how the fitness values are used.

Proportional selection, ranking, and tournament selection are the most popular selection procedures. The procedure of a generic GA [7] is given as follows:

Step 1: Set t ¼ 1. Randomly generate N solutions to form the first population, P1. Evaluate the fitness of solutions in P1.

Step 2: Crossover: Generate an offspring population Qt as follows:

2.1. Choose two solutions x and y from Pt based on the fitness values.

2.2. Using a crossover operator, generate offspring and add them to Qt.

Step 3: Mutation: Mutate each solution xAQt with a predefined mutation rate.

Step 4: Fitness assignment: Evaluate and assign a fitness value to each solution xAQt based on its objective function value and infeasibility.

Step 5: Selection: Select N solutions from Qt based on their fitness and copy them to Pt+1. Step 6: If the stopping criterion is satisfied, terminate the search and return to the current population, else, set t ¼ t+1 go to Step 2.

4. Multi-objective GA

Being a population-based approach, GA are well suited to solve multi-objective optimization problems. A generic single-objective GA can be modified to find a set of multiple non-dominated solutions in a single run. The ability of GA to simultaneously search different regions of a solution space makes it possible to find a diverse set of solutions for difficult problems with non-convex, discontinuous, and multi-modal solutions spaces. The crossover operator of GA may exploit structures of good solutions with respect to different objectives to create new nondominated solutions in unexplored parts of the Pareto front. In addition, most multi-objective GA do not require the user to prioritize, scale, or weigh objectives. Therefore, GA have been the most popular heuristic approach to multi-objective design and optimization problems. Jones et al. [4] reported that 90% of the approaches to multiobjective optimization aimed to approximate the true Pareto front for the underlying problem. A majority of these used a meta-heuristic technique, and 70% of all metaheuristics approaches were based on evolutionary approaches.

The first multi-objective GA, called vector evaluated GA (or VEGA), was proposed by Schaffer [5]. Afterwards, several multi-objective evolutionary algorithms were developed including Multi-objective Genetic Algorithm (MOGA) , Niched Pareto Genetic Algorithm (NPGA), Weight-based Genetic

Algorithm (WBGA), Random Weighted Genetic Algorithm (RWGA), Nondominated Sorting Genetic Algorithm (NSGA), Strength Pareto Evolutionary Algorithm (SPEA), improved SPEA (SPEA2), Pareto-Archived Evolution Strategy (PAES), Pareto Envelope-based Selection Algorithm (PESA), Region-based Selection in Evolutionary Multiobjective Optimization (PESA-II), Fast Nondominated Sorting Genetic Algorithm (NSGA-II), Multi-objective Evolutionary Algorithm (MEA), Micro-GA, Rank-Density Based Genetic Algorithm (RDGA), and Dynamic Multi-objective Evolutionary Algorithm (DMOEA). Note that although there are many variations of multi-objective GA in the literature, these cited GA are well-known and credible algorithms that have been used in many applications and their performances were tested in several comparative studies. Several survey papers have been published on evolutionary multi-objective optimization. Coello lists more than 2000 references in his website. Generally, multi-objective GA differ based on their fitness assignment procedure, elitisim, or diversification approaches. In Table 1, highlights of the well-known multi-objective with their advantages and disadvantages are given. Most survey papers on multi-objective evolutionary approaches introduce and compare different algorithms. This paper takes a different course and focuses on important issues while designing a multi-objective GA and describes common techniques used in multi-objective GA to attain the three goals in multi-objective optimization. This approach is also taken in the survey paper by Zitzler et al. [1]. However, the discussion in this paper is aimed at introducing the components of multi-objective GA to researchers and practitioners without a background on the multi-objective GA. It is also import to note that although several of the state-of-the-art algorithms exist as cited above, many researchers that applied multi-objective GA to their problems have preferred to design their own customized algorithms by adapting strategies from various multiobjective GA. This observation is another motivation for introducing the components of multi-objective GA rather than focusing on several algorithms. However, the pseudocode for some of the well-known multi-objective GA are also provided in order to demonstrate how these procedures are incorporated within a multi-objective GA[8].

5. System Design and Implementation

Our proposed model uses multi objective genetic algorithm for the reduction of test cases. The phases and design of our system is elaborated below in detail.

Statement of the Modeling Problem:

In order to achieve the accurate results for the reduction of test cases it is has become necessary to design and implement the general and efficient system. Web applications have dynamic properties. Testing whole website can be an exhaustive process and you may end up having uncountable test suites so to avoid that we will reduce the test cases to the appropriate amount. It will save the time, cost ad effort also giving the best coverage. The equation given below explains all the steps that will be applied on the ids generated after transactions.

Prioritized test cases=(M(C(S(F(E(v(ip(ids)))))))

Where the symbols are described in the table given below:

Table 1: List of Factors

Symbols	Description
ip	Inputs
ids	Id generated after each transaction
v	Validation
E	Eliminating repeating test cases
F	Fitness
S	Selection
C	Cross over
M	Mutation

Before the start of implementation for such system real world problem needs to be considered. Problems that should be kept in mind while using multi objective genetic algorithm are that which quality attribute to focus for the testing of web application, what will be test adequacy criteria, On what basis fitness will be calculated, which selected test cases to crossover and which bits to mutate in chromosomes after cross over. All these queries are entertained in the proposed model given below.

Proposed solution Algorithm:

```
Initialization of population
Validation
Eliminating redundancy
While (termination condition not met)
{
Fitness Evaluation
Selection
Cross over
Mutation
}
```

Proposed solution Model:

A. Transaction

This phase deals with the input of the system. The web application we have considered for the testing is Yahoo mail beta. When the user will select the certain action ids corresponding to that will be generated. Number of test cases will depend on the transactions that user will make. Each transaction on completion makes a test case. We can consider ids as genes and genes altogether makes a chromosome. To perform any action it is necessary to be logged in. User will first open the yahoo mail page, enter an email address, password and logged in. Ids assigned to such actions are 000, 001, 002 and 003, test case made according to that will be [000, 001, 002, 003].

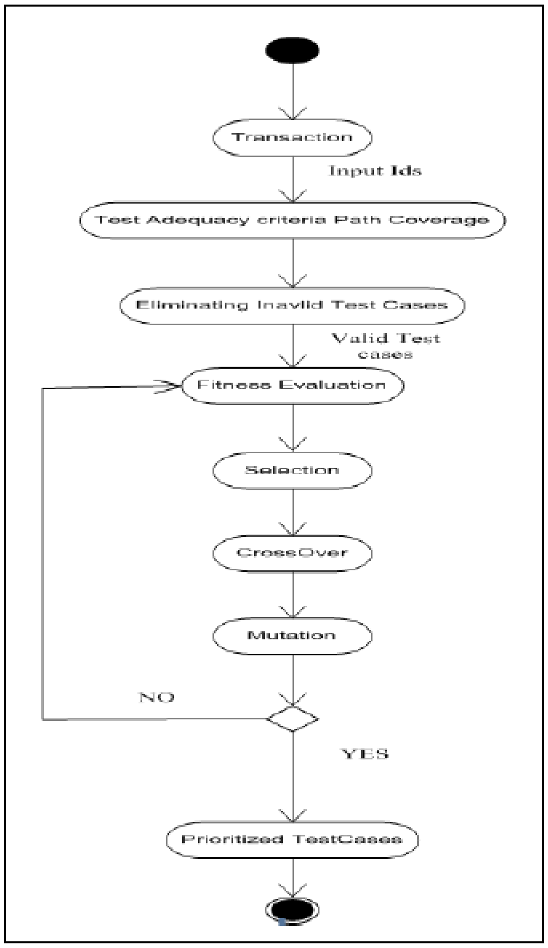

Figure:1 Flow Graph of MOGA

We have assumed this test case to be always true. Other test cases are made with the actions on the yahoo beta home page. Considering one of the other test cases for example to print an email, the respective transactions that user will make will be inbox, select an email, go to actions and print. Test case generated will be [200, 201, 260, 261]. Total actions we are considering from yahoo mail beta home page are 33. These 33 actions make 22 valid test cases.

B. Path Coverage

Path coverage is the test adequacy criteria we have use for the development of our system. It validates the path or test cases user enters. It verifies that whether user has entered the right test cases; if yes it adds it into the population and if not it removes it from the chromosome set. If the test case seems nearer to any valid test cases; but some of it ids are not correct it modifies it into the valid ones. Eliminating invalid test cases from the start helps in efficient reduction. Ai based multi objective genetic algorithm will then only be applied on valid test cases this will for surely produce the best population.

C. Eliminating Repetition:

This step eradicates all the repeating test cases to extract more valid test cases. Repetition can make the population less optimal. If same test cases are being used again and again and all the genetic operations are applied on them, then there is very less probability of producing the best population.

D. Fitness Evaluation:

Fitness calculation is problem specific i.e. it is calculated according to the domain needs. In our model fitness is calculated on the basis of no. of steps. e.g. if u have to send an email, you need to login this is the basic requirement. We have assumed it to be true always.

Once you are logged in you go to compose message, fill the address field and then send. These all transactions total make the length three n this is then multiplied with random number. This gives the fitness of test case send an email. This is how the fitness of all test cases is calculated. If the length of first test case is greater than other test cases then it will have more fitness value. It depends on the number of steps each test case is covering and then given priority according to that.

E. Selection

As the name suggests selection is the process of choosing the best chromosomes. Chromosomes are selected on the basis of the fitness criteria. Chromosomes with greater fitness are considered best and are nominated for the further genetic operations.

F. Cross over

Cross over is the main genetic operation that swaps the half chromosome of the first best parent chromosome with the other best parent and produces the off springs. This is an iterative process and it continues until the best population is found. The criterion we have set for the cross over is on the basis of affinity i.e. the test cases that are somehow familiar with one and other are most likely to crossover. Affinity here suggests similar ids .The test cases with same starting ids will swap with each other. Cross over point is not specific it's up to the developer to set any point.

The cross over point set here is 2. Swapping the chromosomes of similar ids has the probability to produce some valid paths after exchange. If the selected population for cross over is in odd number;

the chromosomes should be converted into even number to make optimal pairs. The chromosomes with the lower length should be discarded as we have the coverage objective to achieve. Chromosome with the larger length will give maximum coverage as it covers more actions. When population is in even number chromosomes with their favorable match ups will make pairs and exchange their ids.

G. Mutation

Mutation brings the biological diversity in the population. If required solution is not being produced from recombination, mutation criteria can be used to bring the change in the produced population and transform it into the desired one. It is not mandatory to perform mutation; it can be omitted if the results are obtained without it.

H. Prioritized Test Cases

Prioritized test cases are the output of the system. These reduced test cases will give the advantage of saving the time, cost of the system and with maximum coverage when testing is done. These reduced test cases are with the valid ids and paths. These are the optimal solutions produced after applying genetic operations.

4. Result and Conclusion

The results gathered after experimentation is as follows. 46 test cases of yahoo mail beta were given as an input to the system and after applying MOGA they were reduced to 18 test cases. 16 of the test cases were invalid; test cases nearer to valid ones were modified. 9 test cases were discarded cause of redundancy and 3 were reduced because they didn't find any pair for cross over. So on the whole 46 test cases were reduced to 18 giving efficient results as it covers the other objectives also. At the end of the implementation the multi objective factors achieved are cost and coverage.

As this project aims to ease the process of testing by reducing the test cases and saving the resources hence it achieves this objective. It saves the cost on testing, as the test cases are prioritized in the end. Fewer test cases to test save the cost and time. Apart from cost the other factor that is achieved is maximum coverage. Test cases are selected on the basis of fitness criteria.

Fitness criteria used in this paper is on the basis of number of steps. Test cases with larger steps are most optimal to be selected. As there are larger steps so it gives the maximum coverage. So on the whole maximum coverage and reduced cost is achieved by prioritizing the test cases.

Although the system made is producing the accurate results with the given criterion but to make it more efficient number of objectives can be enhanced. We are considering cost and coverage; Research with the time factor needs to be covered.

We have chosen the criteria for fitness evaluation i.e. on the basis of length of test case, the other criteria which has the possibility to produce best results is on the basis of actions priority.

Corresponding Author:
Alireza Souri
Department of Computer Engineering,
Ahar Branch, Islamic Azad University,
Ahar, Iran
E-mail: a-souri@iau-Ahar.ac.ir

References

1. Lei Xu, Boawen Xu, "A frame work for web application and testing", International Conference on Cyberworlds, 2004.
2. Myers G. J., The Art of Software Testing, 1st edition, John Wiley and Sons, NY, USA, 1979.
3. Eugene Volokh, VESOFT, "AUTOMATED TESTING -- WHY AND HOW", INTEREX Conference, Published by INTERACT Magazine, Dec 1990.
4. Praveen Ranjan Srivastava, Tai-hoon Kim, "Application of Genetic Algorithm in Software Testing", International Journal of Software Engineering and Its Applications Vol. 3, No.4, October 2009.
5. Christopher C. Michael, Ginput E. McGraw. Michael A. Schatz, and Curtis C. Walton, GA for Dynamic Test. Data Generation**"**, National Science Foundation, 1997/05/23.
6. Holland JH. Adaptation in natural and artificial systems. Ann Arbor: University of Michigan Press; 1975.
7. Goldberg DE. Genetic algorithms in search, optimization, and machine learning. Reading, MA: Addison-Wesley; 1989.
8. Abdollah Konak, David W.Coit, Alice Smite "Multi-objective optimization using genetic algorithms: A tutorial" . Reliability Engineering and System Safety 91 (2006) 992–1007
9. Deb, K. and Kumar, A. (1995). Real-coded genetic algorithms with simulated binary crossover: Studies on multi-modal and multi-objective problems. *Complex Systems*, 9:431–454.
10. Parks, G. T. and Miller, I. (1998). Selective breeding in a multi-objective genetic algorithm. In Eiben, A. E., B¨ack, T., Schoenauer, M. and Schwefel, H.-P., editors, *Proceedings of the Parallel Problem Solving from Nature, V*, pages 250–259, Springer, Berlin, Germany.

3/12/2012