2009

A Max-Min Multiobjective Technique to Optimize Model Based Test Suite

Usman Farooq
Edith Cowan University

Chiou P. Lam
Edith Cowan University

10.1109/SNPD.2009.33 This article was originally published as: Farooq, U. , & Lam, C. P. (2009). A Max-Min Multiobjective Technique to Optimize Model Based Test Suite. Proceedings of International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing. (pp. 569-574). Daegu, Korea. IEEE Computer Society. Original article available here

2009 10th ACIS International Conference on Software Engineering, Artificial Intelligences, Networking and Parallel/Distributed Computing

A Max-Min Multiobjective Technique to Optimize Model Based Test Suite

Usman Farooq, C. P. Lam
School of Computer and Information Science
Edith Cowan University
Perth, Australia
ufarooq@student.ecu.edu.au, c.lam@ecu.edu.au

Abstract— **Generally, quality software production seeks timely delivery with higher productivity at lower cost. Redundancy in a test suite raises the execution cost and wastes scarce project resources. In model-based testing, the testing process starts with earlier software developmental phases and enables fault detection in earlier phases. The redundancy in the test suites generated from models can be detected earlier as well and removed prior to its execution. The paper presents a novel max-min multiobjective technique incorporated into a test suite optimization framework to find a better trade-off between the intrinsically conflicting goals. For illustration two objectives i.e. coverage and size of a test suite were used however it can be extended to more objectives. The study is associated with model based testing and reports the results of the empirical analysis on four UML based synthetic as well as industrial Activity Diagram models.**

Keywords-Model Based Testing; Multobjective Evolutionary Algorithm; Test Suite Optimization; UML;

I. INTRODUCTION

The model-based random testing technique is agile, immune to the pesticide-paradox [2], and characterized by its simplicity and readiness efficacy. While this technique can produce as many test cases as one needs, ironically it can pollute the test suite with an inordinate number of unintentional redundant test cases. A test suite with redundant test cases increases the test suite size, takes far longer to complete without providing any obvious advantage or enhanced confidence. The additional time and effort needed to execute these unwarranted test cases obviously raises the testing cost, diminish overall productivity and waste often-scarce project resources. Moreover, the redundant test case denies the quality of good test case [3].

The process of identifying and discarding redundant test cases that finally yields a minimal test suite can be defined as test suite optimization. As the redundancy of a test case is relative and dependent on the test criteria and other test cases in a test suite, the optimization process may needs to evaluate all possible combinations of the test cases in a test suite and calculate their cumulative coverage. For a test suite with 'n' test cases, the number of evaluations will be the order of 'n' potential combinations. The process of manual identification and removal of redundant test cases is both overwhelmingly complex and erratic. Similarly, exhaustive analysis even with an automated tool would handle only relatively trivial test suites and is deemed impractical for industrial-scale test suites. The complexity of test suite optimization problem is exponentially related to the original test suite size. Thus, because of this combinatorial explosion problem, test suite optimization cannot be attained in polynomial time except for a trivial test suite.

The analogy between the test suite optimization and combinatorial optimization was defined by Harman and Jones [4]. The purpose of applying search-based techniques in software testing is to find optimal solutions for problems where analytical algorithms are either infeasible or too complex owing to huge solution space. Various studies have already demonstrated the prospects of metaheuristic techniques in various software testing problems. Zheng *et al.* (2007) evaluated various search algorithms for regression test case prioritization [5]. Their study concludes that the genetic algorithm is equivalent to greedy algorithms in terms of performance and even more suitable for a situation when the fitness of the test suite is not predetermined. The work presented in this paper is different from their work in two ways. First, our approach aims to minimize the generated test suite and secondly, it is focused on model based testing as compared to code based testing used in their work.

Most of the research reported with the application of metaheuristic techniques is focused on test case prioritization and more specifically code-based techniques [6]. The optimization of test suite is a controversial topic, mainly because of the compromising effects on the fault detect-ability of the reduced test suite. So far, the proposition about the fault detect-ability of optimized test suite is limited to the code-based regression test suite (For more detail please see [7, 8]). One study regarding specification based test suite minimization was conducted but enormous difference between specification techniques and test generation mechanisms indicate that those results are not necessarily applicable to other model based technique. However, this fact highlights the need for further study.

In this paper, we present a novel solution for test suite optimization by means of multiobjective evolutionary

978-0-7695-3642-2/09 $25.00 © 2009 IEEE
DOI 10.1109/SNPD.2009.33

technique. We reformulate the optimization of model-based test suite as a travelling salesman problem with profits (TSPP) [7]. The TSPP is a generalization of the travelling salesman problem (TSP) where the objective is the maximization of profit without the necessity of visiting all cities. To our knowledge, this work is a first attempt to formalize and automate the model-based test suite optimization process with multiobjective evolutionary metaheuristic. We demonstrated the test suite optimization through an example. Empirical study was conducted with industrial scale models and results are compared with those produced by other algorithms. We postulate that a test case is redundant in accordance with a specific criterion if it fails to add extra information or coverage.

The remaining paper is organized as follows. Section 2 describes test suite optimization problem. Multiobjective evolution based test suite optimization technique is introduced in section 3. Experiment, corresponding results and discussion are presented in Section 3.1 and 3.2. Summary and future work is given in Section 4.

II. Test Suite Optimization

Generally, the efficiency of a test suite is measured w.r.t. a particular test requirement metric i.e. structural coverage, fault coverage and mutation score; however here we will use the structural coverage as a surrogate measure of efficiency. We postulate that a test suite is inefficient if it has redundancy (as it will waste resources) and ineffective if it has gaps in the coverage (as it will leave untested functionality in the system). A test case is considered redundant in a test suite in accordance with a specific criterion if it fails to improve consolidated coverage of the test suite. Consider a model-based test suite T contains 'm' test cases, and each test case is a sequence of model elements representing an execution path in the model. These test cases are evaluated with respect to a coverage criterion M that identifies 'n' elements in the artifact under test (AUT). When a subset S of T containing 'x' test case is executed, it traverses 'p' of the 'n' AUT elements and attain 'p/n' percent coverage. To quantify the efficiency and coverage of a test suite, following definitions are stated.

Definition 1: The efficacy of a test suite T w.r.t. a criterion M is a ratio of p to n.

Definition 2: The efficiency of a test suite T w.r.t. a criterion M is a ratio of p to m.

The efficacy metric in definition-1 can be used to measure the effectiveness of a test suite in terms of selected criteria i.e. structural coverage and mutation score. When this value is less than 1 (in terms of ratio) or 100%, it indicates the deficiency in the test suite and the need to generate more test cases. The efficiency metric scales the test suites in terms of execution cost i.e. size and time. The efficiency metric is obvious and the high score indicates relative higher redundancy in a test suite. It is explained as following: Given a test suite TS with test cases t1- t7 to execute a set of seven elements {A, B, …G} in a AUT. The test suite yields efficacy = 1 (complete coverage) and efficiency = 7/6. However, a closer look at the table-1 reveals that the redundancy in a test suite can exist in three forms: 1) a test case duplicates one or more test cases; 2) a test case subsumes the coverage of several test cases and 3) a combination of test cases subsumes the coverage of several test cases. For example in table 1, t7 is a duplication of t4, t7 spares both t3 and t4, and combination of t1 and t5 subsumes the coverage of all other test cases. Skipping the redundant test cases can improve the test suite efficiency e.g. the subset TS_{sub}={t1,t5} of TS can improve the efficiency from 7/6 to 7/2 without compromising its efficacy.

TSP is one of the most intensely studied problems in combinatorial optimization. In the basic version of the TSP, the objective is to find a most cost effective round trip path for a given number of cities with traveling cost between them. Profitable Tour Problem (PTP) is a multiobjective type of TSP with profits (TSPP) and formally defined as follows [9]. Consider G = (V, E) be a complete undirected graph where V = {v_1, v_2, . . . , v_n} is a set of n vertices and E is a set of edges. A profit p_i is associated with each vertex v_i ∈ V (with $p_1 = 0$) and a distance c_{ij} with each edge (v_i, v_j) ∈ E. The objective is to find a tour with two conflicting objectives: (1) minimize the travel cost and (2) maximize the total profit.

Table 1: Example test suite with coverage illustration

	A	B	C	D	E	F	G
t1	x		x	x	x		
t2	x	x		x	x		x
t3	x		x			x	
t4	x	x	x	x		x	
t5	x	x		x		x	x
t6	x		x		x	x	x
t7	x	x	x	x		x	

In analogy to the PTP, the model-based test suite optimization problem can be defined as finding a subset of test suite TS with minimum number of test cases and maximum coverage. For instance, the test suite has 'n' test cases that correspond to cities in PTP. Each test case 'i' has coverage that corresponds to the profit 'p_i' associated with a city. Like PTP, the selection of a city to visit, a binary variable 'y_i' is used to indicate the inclusion or exclusion of a test case. The traveling cost 'c_e' associated with an edge in PTP implies the utility value of a test case and a binary variable 'x_e' associated with the edge indicates if the corresponding edge is used in the solution or not. As the

objective is to find a subset of test suite TS with maximum coverage at minimum cost, formally the problem can be stated as [9]:

$$minimize \sum_{e \in E} c_e x_e - \sum_{v_i \in V} p_i y_i \quad (1)$$

The requirements to be satisfied are

$$\sum_{e \in \delta(\{v_i\})} x_e = 2y_i \quad (v_i \in V) \quad (2)$$

$$subtour\ elimination\ constraints, \quad (3)$$

$$y_1 = 1, \quad (4)$$

$$x_e \in \{0,1\} \quad (e \in E) \quad (5)$$

$$y_i \in \{0,1\} \quad (v_i \in V) \quad (6)$$

III. Multiobjective Evolution based Test Suite Optimization

Evolutionary Computation (EC) is a metaheuristic, inspired by the natural process of evolution. Generally, the EC algorithms are designed to optimize only a single objective or decision variable. However, most of the real world's optimization problems have more than one decision parameter involved and often the good tradeoffs are searched for competing constraints. For these problems, usually more than one equally good solution exists and choosing the best one always depends upon the application context. However with single-objective EC algorithms, these problems are optimized with only one objective while others are handled as constraints. In multiobjective algorithms, all objectives are optimized simultaneously. Formally, the multiobjective optimization can be stated as follows:

$$minimize\ F(x) = \left(f_1(x), \ldots, f_n(x)\right) \quad (7)$$

$$subject\ to \quad x \in D \quad (8)$$

where F(x) is the vector of objectives; $n \geq 2$ is the number of objective functions; $x = (x_1, x_2, \ldots, x_r)$ is the vector of decision variables; and D is the feasible solution space.

Generally, the multiobjective evolution based algorithms (MOEA) are classified according to the underlying solution propagation mechanism, i.e. pareto and non-pareto based techniques. The pareto based techniques e.g. NGPA, NSGA and SPEA, produce multiple distinct solutions a.k.a. pareto optimal (solution) set at each iteration. However, the non-pareto class of techniques i.e. VEGA and Min-Max, designed to propagate a global unique solution. For more details about pareto and non-pareto based techniques, see [10, 11].

For test suite optimization, we adapted a non-pareto based min-max technique [12] and called it multiobjective evolutionary technique MOET. However, other MOEA techniques could be chosen instead as well. The incorporation of min-max approach needs the formulation of a fitness function based on the goal points to guide the underlying heuristic search. The goal points are the upper bounds of the desired objectives. For instance, with coverage objective the fitness of the test suite fitness is computed as per its proximity to the maximum coverage. The maximum test suite size is relative to the search space which means solution subset in worst case could be equal to the original test suite. So, for size objective the fitness of the test suite is based on its deviation from the maximum test suite size. Similar to the min-max dominance strategy, we devise an annealing weight function for coverage objective relative to the distance from that goal point. Following is the expression of objective function:

$$L_p(f) = \sum_{i=1}^{n} w_i \left| \frac{f_i^0 - f_i(x)}{\rho_i} \right| \quad (9)$$

where n is the number of objectives; w_i is the weight associated with each objective; and depending on which gives maximum value for $L_p(f)$, either of the f_i^0 and $f_i(x)$ will be used for ρ_i.

The incorporation of the evolution based algorithm needs a number of careful design decisions i.e. fitness function, offspring generation and selection mechanism, and then further refinement is needed in parametric values to balance the convergence and divergence of the search. For parent selection, tournament selection was preferred over roulette-wheel and elitism due to its proven superior performance. Crossover is a mechanism to combine parent to produce new solution. As we are using steady-state version so only offspring are produced. Offspring are randomly modified according the given mutation rate. Initially, the population is randomly generated and the parametric values are used as suggested in [13]. However, the parametric values are then refined to improve the performance of the algorithm. The final parametric values for the MOET are mentioned in the table 2.

Table 2: Parametric settings for the MOET

Parameters	Value
Objective 1	Minimize test suite size
Objective 2	Maximize test suite coverage
Population size	50
No. of Generations	500
Replacement scheme	Steady state
Crossover rate	0.9 (double point)
Mutation rate	0.2 (single point)
Tournament size	10
Multiobjective strategy	Min-max

A. Experiment

In order to investigate the feasibility of multiobjective optimization of model based test suite, we conducted the experiment with four models of various sizes and complexity levels. The AD model shown in fig. 1, describes an Enterprise Customer Commerce System (ECCS) taken from [1]. It describes the process of online purchasing of products that is comprised of two sub-processes: authentication and shopping. The first process authorizes the existing users for shopping and configuration. However, in case of new customers, it enables them to register first. The shopping process facilitates the user to order the selected products and configure his/her account if required. The Automatic Teller Machine (ATM) model is a popular case study. For our experiment, we adapted it from a report [14]. The ATM model comprises on an activity diagram with top level view of ATM operations that is further decomposed into low level AD diagrams with details of the operations i.e. withdraw cash, deposit money, transfer funds and check balance. The experiment also includes the two industrial scale models of a module in an Intelligent Transport System (ITS), Edit Trend Properties (ETP) and Delete Trend Properties (DTP). Both models respectively describe the step by step editing and deletion of existing trending reports from archived or real-time data. We use the model (ECCS) as a running example. It is a small but not simple as its structural cyclomatic complexity is 11, which is higher than ATM and DTP models. For more detail of each model see table 1. These models are used in the experiments and results are presented in the following sections.

Test suites are generated for all models using a Random-walk based test sequence generation (TSG) algorithm proposed in [15]. For example, 20 test cases are generated for ECCS model. Given the stochastic nature of the TSG technique, it is assumed that the generated test suites incurred few redundant test cases. The generated sequences of model constructs, formally referred to as paths, are usually evaluated according to a specified criterion. An AD based branch coverage criteria defined in [15] is used to evaluate and optimize the test suite as it is minimum requirement for software in aviation industry. The branch coverage criterion requires that there would be at least one test case for each branch that causes it to execute.

Table 3: Characteristics of sample models

Model	Nodes	Branches	Edges	Complexity
ECCS	23	17	33	11
ATM	135	28	141	16
ETP	77	26	89	14
DTP	52	37	57	21

The heart of the Evolution based technique is the fitness function that calculates the fitness of an individual test suite w.r.t. a given criterion by obtaining its phenotypic properties (i.e. coverage and size) by mapping it on the search space of the randomly generated test cases. In analogy to the PTP, the multiobjective optimization for model-based test suite is formulated as:

Find a subset of the test suite which satisfy the constraints and optimize following criteria:

1. Size of the subset by removing redundant test cases; and
2. Coverage of the subset w.r.t. given branch criterion.

The first objective is to minimize the test suite size while the second objective is to maximize the coverage. The vector of decision variables is x = [size, coverage].

Given the fact that the upper bound of the coverage C_{UB} that a subset (solution test suite) can attain w.r.t. to a particular criterion is 100%, and the upper bound for a subset size S_{UB} is the size of original test suite so the objective functions can be expressed as follows:

1. Size: $f_1(x) = S_{UB} - S_x$
2. Coverage: $f_2(x) = C_{UB} - C_x$

For analysis, the performance of the multiobjective technique proposed in this paper is compared with that of single objective GA w.r.t. branch coverage criterion. For single objective GA (SGA), the test suite optimization problem is transformed into Quota Traveling Salesman Problem (QTSP) which is another variant of PTP [8]. In QTSP, the minimum allowed profit is imposed as a bound and the goal is to find a minimal length tour whose total collected profit is not smaller than this bound. We specified the minimum coverage as a lower bound and problem is defined to find a subset with minimal number of test cases and the total coverage of the subset is not smaller than the lower bound.

The results of the experiments and discussion are presented in the next section. Given the stochastic nature of the optimization algorithm, a small variation is expected in the results. Therefore, experiment is designed to run for 10 iterations for each criterion to determine the stability and effectiveness of the optimization framework.

B. Results and Discussion

The results of the experiments are presented in table 4 and 5. The size column shows the test suite size and efficiency column is calculated according to definition 2 specified in section 2. The efficiency of un-optimized and optimized test suites is calculated by 100% coverage of the branches given in table 3 and size of un-optimized and optimized test suite respectively. The efficiency in table 5 is calculated in similar fashion, however, the number branches traversed by the specific test suite is used with the size of that test suite.

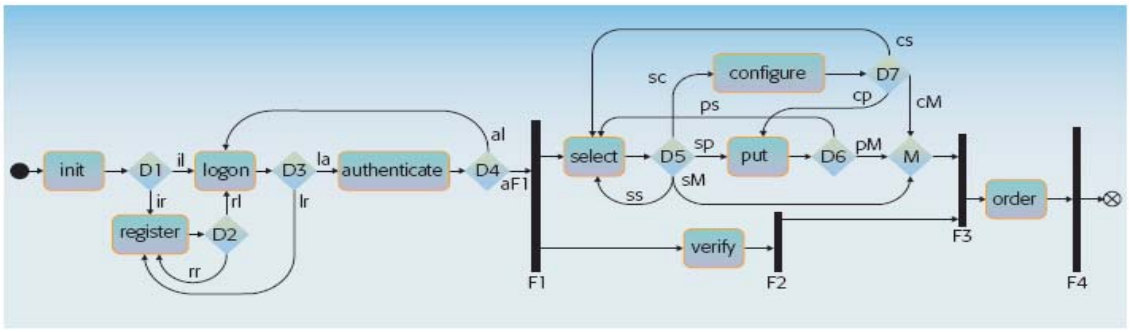

Figure 1: AD model of an Enterprise Customer Commerce System (ECCS)[1]

Table 4: Coverage maximization and Size Reduction w.r.t. Branch Criterion (Test suite with complete coverage)

Model	Un-Optimized TS		Optimized TS		
	Size	Efficiency	Algorithm	Size (reduction%)	Efficiency
ECCS	19	17/19	SGA	3 (85.0%)	17/3
	19	17/19	MOET	3 (85.0%)	17/3
ATM	89	28/89	SGA	7 (92.1%)	28/7
	89	28/89	MOET	7 (92.1%)	28/7
ETP	27	26/27	SGA	6 (77.8%)	26/6
	27	26/27	MOET	6 (77.8%)	26/6
DTP	28	37/28	SGA	6 (78.6%)	37/6
	28	37/28	MOET	7 (75.0%)	37/7

Table 5: Test Suite Optimization w.r.t. Branch criterion (Test suite with less than 100% coverage)

Model	Un-Optimized TS		Optimized TS		
	Size	Efficiency	Algorithm	Size (reduction%)	Efficiency
ECCS	13	17/13	SGA	3 (76.9%)	16/3
	13	17/13	MOET	3 (76.9%)	16/3
ATM	62	28/62	SGA	7 (88.7%)	27/7
	62	28/62	MOET	7 (88.7%)	27/7
ETP	25	26/25	SGA	5 (80.0%)	21/5
	25	26/25	MOET	5 (80.0%)	21/5
DTP	25	37/25	SGA	5 (80.0%)	36/5
	25	37/25	MOET	5 (80.0%)	36/5

The results exhibit two important phenomena: 1) the removal of redundant test cases improves the test suite efficiency without compromising its efficacy; and 2) both SGA and MOET are competitive in terms of performance. From data, the test suite efficiency is quite obvious. As it can be seen that the reduction in test suite size of each example model is quite significant Although, there is no significant difference in both SGA and MOET, however, the MOET has the advantage that it can optimize multiple objectives in parallel. The result confirms the feasibility of the multiobjective optimization of the model based test suites. As in the experiment, four different models were used. The test suite generated for these models also varies in terms of size and redundancy. However, both SGA and MOET performance proved robust and scalable.

IV. SUMMARY

The field of search-based software engineering is new and the incorporation of various metaheuristic techniques has heralded a new era of research and development. In software testing and particularly in structural testing several researchers have successfully incorporated these techniques for test data generation to regression test suite prioritization. However, in terms of model based testing still much need to be done. In this paper we proposed a multiobjective optimization framework for model based test suites. We formulated the test suite optimization as travelling sales man with profit problem. A multiobjective min-max technique was implemented with evolutionary technique and results are compared with classical single objective GA. The

experimental results show the robustness of the proposed technique that optimizes the test suites generated from AD model w.r.t. the branch and edge coverage criterion. The study confirms the test suite optimization by eliminating the redundant test cases without compromising its coverage.

REFERENCES

[1] J. M. Küster, J. Koehler, and K. Ryndina, "Improving Business Process Models with Reference Models in Business-Driven Development," presented at Business Process Management Workshops, 2006.

[2] B. Beizer, *Software testing techniques*, 2nd ed. New York: Van Nostrand Reinhold, 1990.

[3] C. Kaner, J. Falk, and H. Q. Nguyen, *Testing Computer Software* 2nd ed: International Thomson Computer Press, 1993.

[4] M. Harman and B. F. Jones, "Search based software engineering.," *Information and Software Technology*, vol. 43, pp. 833-839, 2001.

[5] Z. Li, M. Harman, and R. Heirons, "Search Algorithms for Regression Test Case Prioritisation " *IEEE Transactions on Software Engineering.* , vol. 33, pp. 225-237, 2007.

[6] M. Harman, "The Current State and Future of Search Based Software Engineering," in *Int. Conference on Software Engineering, Future of Software Engineering (FOSE'07)*. Minneapolis, USA: IEEE, 2007.

[7] W. E. Wong, J. R. Horgan, L. London, and A. P. Mathur, "Effect of Test Set Minimization on Fault Detection Effectiveness," presented at 17th International Conference on Software Engineering, 1995.

[8] G. Rothermel, M. J. Harrold, J. Ostrin, and C. Hong, "An Empirical Study of the Effects of Minimization on the Fault-Detection Capabilities of Test Suites," in *International Conference on Software Maintenance (ICSM 1998)*. Bethesda, MD, 1998.

[9] F. Dominique, D. Pierre, and G. Michel, "Traveling Salesman Problems with Profits," *Transportation Science*, vol. 39, pp. 188-205, 2005.

[10] E. Zitzler, "Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications," in *Computer Engineering and Networks Laboratory*, vol. Doctrate. Zurich: Swiss Fedral Institute of Technology, 1999, pp. 118.

[11] C. A. C. Coello, "An updated survey of GA-based multiobjective optimization techniques," *ACM Comput. Surv.*, vol. 32, pp. 109-143, 2000.

[12] C. A. C. Coello and A. D. Christiansen, "Two New GA-based methods for multiobjective optimization," *Civil Engineering and Environmental Systems*, vol. 15, pp. 207-243, 1998.

[13] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*: Addison-Wesley, 1989.

[14] R. Chandler, C. P. Lam, and H. Li, "UML Models with Activity Diagrams: for Case Studies," SCIS, Edith Cowan University, Perth, Technical Report TR-SERG-06-02, 2006.

[15] U. Farooq, C. P. Lam, and H. Li, "Towards Automated Test Sequence Generation," presented at 19th Australian Software Engineering Conference (ASWEC'2008), Perth, Australia, 2008