# Multi Objective Algorithms for Automated Generation of Combinatorial Test Cases with the Classification Tree Method

Peter M. Kruse[1] and Kiran Lakhotia[2]

[1] Berner & Mattner Systemtechnik GmbH, Gutenbergstr. 15, Berlin, Germany
peter.kruse@berner-mattner.com,http://www.berner-mattner.com/
[2] CREST, UCL, Gower Street, London, UK
k.lakhotia@cs.ucl.ac.uk,http://crest.cs.ucl.ac.uk/

**Abstract.** Test case selection and prioritization are well studied and understood regression testing techniques. Equally, test case generation is an active research area. Yet the combination of these techniques remains largely unexplored. This paper proposes to use a multi objective approach to combine a test case generation technique, the Classification Tree Method, with a test case selection and prioritization method. Our work aims to generate optimized test suites, containing test cases ordered according to their importance with respect to test goals. We plan to incorporate the algorithms we develop during this work into the Classification Tree Editor, an industrial strength testing tool provided by Berner & Mattner, and will be empirically evaluating our approach on a set of benchmark systems.

**Keywords:** classification tree method, prioritized test case generation, test case selection, test suite optimization

## 1 Introduction

In an ideal world a tester would be able to run as many tests as required. However, due to limited resources, such as manpower, availabililty of test infrastructure like Hardware In the Loop systems *etc.*, a tester often has to select a subset of test cases to run. Despite this, a tester will aim to maintain a sufficient level of test coverage and failure detection capability. If high-value test cases have been determined from previous software projects, a tester could start with those and supplement the tests with additional test cases from a global test case list for the new project. While there is no guarantee the selected test cases are more important than any of the other test cases, there is no well-established way to identify the most important test cases from a given test suite.

In this article we propose to investigate the use of multi objective algorithms in order to combine a test case generation technique, the Classification Tree Method, with a test case selection and prioritization method. The objective of the proposed work is to generate optimized test suites, containing test cases ordered according to their importance with respect to test goals. Yoo and Harman [1]

have already shown that multi objective algorithms can be applied to test case selection and prioritization problems. In 2011, Harman [2] gave an invited talk at a regression workshop in which he further underlined the need for multi objective algorithms to tackle the kind of problems which we will also be considering in this paper.

## 2 Definitions and Background

### 2.1 Combinatorial Interaction Testing

Combinatorial Interaction Testing (CIT [3]) is an effective testing approach for detecting failures caused by certain combinations of components or input values. The tester identifies the relevant test aspects and defines corresponding classes. These classes are called *parameters*, their elements are called *values*. We assume the parameters to be disjoint sets. A test case is a set of $n$ values, one for each parameter.

CIT is used to determine a smallest possible subset of tests that covers all combinations of values specified by a coverage criterion with at least one test case. A coverage criterion is defined by its strength $t$ that determines the degree of parameter interaction and assumes that all parameters are considered.

The most common coverage criterion is 2-wise (or *pairwise*) testing, that is fulfilled if all possible pairs of values are covered by at least one test case in the result test set. A large number of CIT approaches have been presented in the past. An overview and classification of approaches can be found in [4] and [5], while [6] provide a recent survey of CIT and its evolution. A survey that focuses on CIT with constraints is given in [7]. Nearly all publications investigate pairwise combination methods, but most of them can be extended to arbitrary $t$-combinations.

Elbaum et al. provide a good overview of existing prioritization approaches [8]. There are two known algorithm supporting prioritized test case generation. The first is an algorithm published in [9], which is an extension to [10]. For efficiency reasons, this algorithm does not consider constraints [7].

The other one is a Binary Decision Diagram (BDD)-based approach published in [11]. The idea is to build a single BDD for a test problem taking constraints into account. The BDD then holds all valid assignments and is used to successively read valid test cases from it in order of their importance.

### 2.2 Classification Tree Method and Classification Tree Editor

The classification tree method [12] aims at systematic and traceable test case identification for functional testing over all test levels (for example, component test or system test). It is based on the category partition method [13], which divides a test domain into disjunctive classes representing important aspects of the test object. The parameters in a classification tree method are called *classifications* and their values are called *classes*.

The Classification Tree Editor (CTE XL) was introduced together with the classification tree method [12]. Current versions of the CTE XL support automated test case generation using CIT and user-defined constraints, so called dependency rules [14]. Current test case generation offers four different coverage modes: *Minimal combination* creates a test suite that uses every class from each classification at least once in a test case. *Pairwise combination* creates a test suite that uses every class pair from disjunctive classifications at least once in a test case. *Threewise combination* ("triple-wise") creates a test suite that uses every triple of classes from disjunctive classifications at least once in a test case. *Complete combination* creates a test suite that uses every possible combination of classes from disjunctive classification in a test case.

Prioritization is used to assign values of importance to classification tree elements. This is typically done by a test engineer prior to test case generation. These values of importance are called *weights*. To cover various kinds of test aspects, these weights can differ. Higher and lower weights reflect higher and lower importance, respectively. Consequently, one is able to compare the elements of the classification tree to determine their importance under a given test aspect and to guide test case generation by priorities.

An introduction to prioritized test case generation with the classification tree method can be found in [15].

### 2.3 Multi Objective Algorithms

In Multi Objective Problems (MOP), algorithms have to optimize two or more conflicting constraints. A typical example of a MOP is the Knapsack problem, where the goal is to minimize the weight of a sack while maximising the profit (by placing items into the sack). MOPs do not usually have a single solution. Instead, a decision maker has to find a good trade-off between the different objectives. This is achieved by generating a set of *Pareto optimal* solutions. Such a set contains only *nondominating* solutions and its representation in the objective space is called a *Pareto front*. The concept of domination is defined as:

> Individual X dominates Y if, and only if, X is better than Y in at least one objective, and no worse in all other objectives.

More formally, assuming the goal is to minimize all objectives, a Pareto front and Pareto optimal set can be defined as ([16]):

> **Pareto Optimal Set:** For a given MOP $\boldsymbol{f}(\boldsymbol{x})$, the Pareto Optimal Set $(P*)$ is defined as: $P* := \{\boldsymbol{x} \in F \mid \neg\exists \boldsymbol{x'} \in F\ \boldsymbol{f}(\boldsymbol{x'}) \leq \boldsymbol{f}(\boldsymbol{x})\}$, where $F$ is the decision variable space.

A subset of Evolutionary Algorithms, known as Multi Objective Evolutionary Algorithms (MOEA), are naturally suited for MOPs. Firstly, they are population based. Some individuals within the population will be better suited for one objective than another and so the population evolves over time into a Pareto

optimal set. The second advantage of MOEAs is their ability to optimize a MOP in a single run. Alternative methods, such as hill climbers or a single objective Genetic Algorithm, can only find a single point on a Pareto front in any given run. Therefore these methods need to be executed repeatedly in order to explore the full Pareto front. In addition, for each run, weights for different objectives have to be adjusted so that the search is able to find alternative solutions of a Pareto front. However, deciding on a good set of weights is not an easy problem.

Over the years researchers have come up with many different MOEAs. This paper will only consider two well known algorithms: NSGA-II [17] and SPEA-II [18].

**NSGA-II** is an Elitist Nondominated Sorting Genetic Algorithm. After evaluation, individuals are divided into different fronts. The first front contains only solutions that are not dominated by any other individuals. It represents the current Pareto front. Once it has been generated, the solutions are (temporarily) removed from the population and the process is repeated for the remaining individuals, generating the second frontier and so forth.

After all individuals have been assigned to a frontier, every individual within a front is assigned a *crowding distance*. The crowding distance measures the average distance of two solutions on either side of an individual, along each of the objectives. Given two individuals within the same frontier, one individual is preferred over another if it has a greater crowding distance. It means the individual lies in a less crowded region of the Pareto front, helping the optimization to create a more diverse set of solutions.

Once all members of a population have been assigned to a frontier and been given a crowding distance, the entire population can be ranked. The NSGA-II then uses a ranked based selection strategy to generate the mating population.

**SPEA-II**, a Strength Pareto Evolutionary Algorithm, maintains a separate fixed size archive in addition to the population being optimized. After every evaluation step, all nondominated solutions are copied into the archive. At every update, the archive is pruned of solutions that have become dominated by new additions. Once the archive is full, further solutions are removed based on a clustering technique, to make room for new solutions in successive iterations. The clustering aims to preserve the characteristics of the current Pareto front despite removing solutions.

In SPEA-II, individuals in a population are assigned a fitness value based on the number of solutions they dominate and are dominated by, both within the current population and the archive population. This is called the *strength* of an individual. In case two individuals are of equal strength, SPEA-II uses a density measure to distinguish between solutions. This measure is an adaptation of the k-th nearest neighbour method [19].

## 3 Proposal

For the usage of search-based techniques in the CTE XL, we see three consecutive fields:

**1.** The search for optimal (as small as possible) $t$-wise coverage for any given test problem: As shown in the previous section, there is a lot of existing work, from Colbourn [9, 10], Cohen [3, 7], and others [20]. This includes using search-based approaches like simulated annealing to achieve a small $t$-wise coverage. We want to enhance existing approaches and make them available for the Classification Tree Method. CTE XL currently uses a non-search-based approach to generate $t$-wise coverage. As there are many benchmarks available, our results will be easily comparable, concerning both effectiveness and efficiency.

**2.** The search for optimal prioritized $t$-wise coverage: As introduced by [9], the best weight coverage in early test cases is an important field of research. CTE XL supports prioritized test case generation, although it uses a deterministic approach. Very latest work by Segall et al. about the FoCuS Tool [11] also targets this (and the previous) problem. However they too do not use a search-based approach. We plan to use a multi objective algorithm to try and find the smallest test set for $t$-wise coverage while at the same time prioritizing test cases generation to maximise weight coverage in early test cases. Benchmarks are available [9, 11], so we can compare our results with other work.

**3.** The search for optimal (as small as possible) test sequences via generation rules: There is not real previous work here, we are looking for something like "parallel chinese postman" / "orthogonal chinese postman" or similar techniques. Initial work [20] considers test sequence generation, but to the best of our knowledge, no work supports parallel test sequence generation.

We want to use different search-based techniques to solve all three problems. As there are existing algorithms for problems 1 and 2, we will be able to compare our results with other approaches (both search-based and conventional). For the third problem, we will need to identify a good set of benchmarks first.

## 4 Conclusions

In this paper, we identified three problems for future research. We aim to compare our approach to existing techniques (both search-based and conventional) for combinatorial test design. We will implement our algorithms in the Classification Tree Editor and perform an empirical study using a set of benchmark systems to evaluate the proposed approach.

**Acknowledgments** This work is supported by EU grant ICT-257574 (FITTEST).

## References

1. Shin Yoo and Mark Harman. Using hybrid algorithm for pareto effcient multi-objective test suite minimisation. *Journal of Systems Software*, 83(4):689–701, April 2010.
2. Mark Harman. Making the case for morto: Multi objective regression test optimization. In *The 1st International Workshop on Regression Testing (Regression 2011)*, Berlin, Germany, 2011.
3. Myra B. Cohen, Joshua Snyder, and Gregg Rothermel. Testing across configurations: implications for combinatorial testing. *SIGSOFT Softw. Eng. Notes*, 31:1–9, November 2006.

4. Mats Grindal, Jeff Offutt, and Sten F. Andler. Combination testing strategies: a survey. *Softw. Test., Verif. Reliab.*, 15(3):167–199, 2005.
5. Rick Kuhn, Yu Lei, and Raghu Kacker. Practical combinatorial testing: Beyond pairwise. *IT Professional*, 10(3):19–23, 2008.
6. Changhai Nie and Hareton Leung. A survey of combinatorial testing. *ACM Comput. Surv*, 43(2):11, 2011.
7. Myra B. Cohen, Matthew B. Dwyer, and Jiangfan Shi. Interaction testing of highly-configurable systems in the presence of constraints. In *Proc. of the 2007 International Symposium on Software Testing and Analysis*, pages 129–139, New York, NY, USA, 2007. ACM.
8. S. Elbaum, A.G. Malishevsky, and G. Rothermel. Test case prioritization: A family of empirical studies. *IEEE Transactions on Software Engineering*, 28(2):159–182, 2002.
9. Renée C. Bryce and Charles J. Colbourn. Prioritized interaction testing for pairwise coverage with seeding and constraints. *Information & Software Technology*, 48(10):960–970, 2006.
10. Charles J. Colbourn and Myra B. Cohen. A deterministic density algorithm for pairwise interaction coverage. In *Proc. of the IASTED International Conference on Software Engineering*, pages 242–252, 2004.
11. Itai Segall, Rachel Tzoref-Brill, and Eitan Farchi. Using binary decision diagrams for combinatorial test design. In *Proc. of the 2011 International Symposium on Software Testing and Analysis*, New York, NY, USA, 2011. ACM.
12. Matthias Grochtmann and Klaus Grimm. Classification trees for partition testing. *Softw. Test., Verif. Reliab.*, 3(2):63–82, 1993.
13. T. J. Ostrand and M. J. Balcer. The category-partition method for specifying and generating fuctional tests. *Communications of the ACM*, 31(6):676–686, 1988.
14. Eckard Lehmann and Joachim Wegener. Test case design by means of the CTE XL. *Proc. of the 8th European International Conference on Software Testing, Analysis and Review*, December 2000.
15. Peter M. Kruse and Magdalena Luniak. Automated test case generation using classification trees. *ASQ Software Quality Professional*, 13:4–12, December 2010.
16. Arturo Hernández Aguirre, Salvador Botello Rionda, Carlos A. Coello Coello, Giovanni Lizárraga Lizárraga, and Efrén Mezura Montes. Handling Constraints using Multiobjective Optimization Concepts. *International Journal for Numerical Methods in Engineering*, 59(15):1989–2017, April 2004.
17. K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE-EC*, 6:182–197, April 2002.
18. Eckart Zitzler, Marco Laumanns, and Lothar Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. In *Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems*, pages 95–100, Athens, Greece, 2002.
19. Bernard. *Density Estimation for Statistics and Data Analysis (Chapman & Hall/CRC Monographs on Statistics & Applied Probability)*. Chapman and Hall/CRC, 1 edition, April 1986.
20. D. Richard Kuhn, Raghu N. Kacker, and Yu Lei. Practical combinatorial testing. Technical report, National Institute for Standards and Technology (NIST), October 2010.