8. Future Directions

Combinatorial testing has evolved into an accepted practice in software engineering. As it has entered mainstream use, research interest has become more specialized and application-oriented. Progress continues to be made in covering array generation algorithms, often with the aim of applying combinatorial methods to a broader range of testing problems, particularly those with larger inputs and complicated constraints. Concurrently, researchers are improving combinatorial test design methods by focusing on input model analysis and tools to assist in this phase of test design. Combinatorial testing continues to expand into domains such as Software Product Lines and mobile applications. Here we review current and upcoming developments in these areas, and suggest potential impacts for practical testing. Finally, we briefly discuss harder problems in the field for which broadly effective solutions are not fully perfected.

8.1 Algorithms

While conventional algorithms produce very compact arrays for many inputs, improvements are being achieved. One recent trend in covering array algorithms is the use of reduction strategies on existing arrays. That is, a t-way covering array with N tests is systematically reduced to fewer than N tests using a variety of mathematical transformations. The near-term impacts of algorithm improvements in array construction include extending the applicability of combinatorial methods. For applications such as modeling and simulation, where a single test may run for hours, reducing covering array size by even a few tests is of great value.

These methods have recently improved upon the best-known sizes of some covering array configurations [1, 2] and active research continues in this area. Similar transformations can also be done where there are constraints, and if the existing test suite was not designed as a covering array [3], using reductions that preserve the combinatorial coverage of the original test suite. An extension of this strategy [4] includes the option of allowing a subset of parameters to have values freely assigned, i.e., new tests can be generated rather than requiring them to be selected from the original test set. Other work shows that heuristic search can in some cases compete with greedy methods in speed and practicality for covering array construction [6]. Additionally, greedy algorithms can be improved using graph-coloring methods [7], to improve on a covering array generation phase that is optimal for t=2 but does not retain optimal properties at t>2.

A somewhat different aspect of applying combinatorial methods in test suite reduction is the use of interaction coverage as a criterion for reducing a test suite [8]. This may be particularly valuable for regression testing. Various test reduction strategies have been applied in the past, but sometimes result in deteriorating fault-detection effectiveness. Since combination coverage is effective in fault detection, retaining high combinatorial coverage in a reduced test set can preserve effectiveness using fewer tests. Yet another practical consideration is the setup time between tests. Many testing problems, especially for system integration or other large system tests, require changes to the SUT configuration with each test. Minimizing this time, while retaining high combination coverage can thus be an effective strategy [5].

8.2 Input Modeling

A second major research trend involves the integration of combinatorial methods in the development environment, and addressing practical problems particular to various domains. The first step in any testing effort is to understand and define the input model, that is, the set of parameters and values that will be included in tests, along with any constraints on values or sequencing. This phase is an issue for any testing approach, not just combinatorial, but the unique aspects of CT have led researchers to tailor conventional methods. Test environments tailored to CT are being developed [9, 10] to work with

popular frameworks such as Eclipse. These environments will allow for validating the consistency and other meta-properties of constraint sets [11].

Software product lines are increasingly used and their enormous range of possible configurations provides a natural domain for combinatorial testing. An extensive survey [16] shows the variety of ways in which *t*-way testing is now being applied in SPL testing and evaluation. Because of the large number of parameters in many SPLs, methods are being devised to extend the range of practical application for covering array generators. Software product lines often have hundreds, or even thousands, of variables. Conventional covering array algorithms are resource-limited in both time and storage to a few hundred. One approach is flattening of the input models, as described in Sect.7.5 [13]. Such methods are an active area of research.

Two current lines of research for improving definition of the input model are classification trees and UML models. UML sequence diagrams can be used as inputs to rule-based tools that extract an input model that can be used with a covering array generator [12]. Input variables and values are extracted from UML message specifications and guard conditions, providing partial automation of the process to reduce effort for test designers. Classification trees fit well with *t*-way testing, because they allow easy analysis and definition of test parameters in a tree structure [14]. Leaf nodes of the tree can be treated as category partitions and used directly in generating covering arrays. Robust tools based on classification trees, UML diagrams, and related concepts can help make combinatorial methods easier to use for test developers.

8.3 Harder problems

Combinatorial testing will continue to find new domains of application, but some research problems remain to be solved. Two broad areas in particular are likely to receive attention from researchers, because of their practical significance in industrial applications.

Very large systems: As with many areas of software engineering, *scalability* is essential. Fortunately, current combinatorial methods and covering array generators can address the vast majority of testing requirements. As noted earlier in the chapter, however, development approaches such as software product lines may involve thousands of parameters, with large numbers of constraints. Current covering array algorithms do not scale to such large problems, and existing constraint solvers are also insufficient for an extremely large number of constraints and variables.

Test development time: Case studies and experience reports show that combinatorial methods can provide better testing at lower cost, but these methods can require significant expertise and do not necessarily speed up the testing process. As such, if time-to-market is the primary concern, conventional test methods are likely to be preferred by developers. Application domains where CT has seen the most rapid acceptance so far are those with very high assurance requirements, such as aerospace/defense, finance, and manufacturing. Reducing the time required for using combinatorial methods is a significant challenge.

Research and practice have shown that combinatorial testing is highly effective across a range of testing problems, and this range of applicability continues to expand for new domains and technologies. The current high level of research interest in the field suggests that it may continue to advance, providing stronger testing at reduced cost for developers.

Additional references

1. Avila-George, H., Torres-Jimenez, J., Gonzalez-Hernandez, L., & Hernández, V. (2013). Metaheuristic approach for constructing functional test-suites. *IET software*, *7*(2), 104-117.

2. Li, X., Dong, Z., Wu, H., Nie, C., & Cai, K. Y. (2014, March). Refining a Randomized Post-optimization Method for Covering Arrays. In *Software Testing, Verification and Validation Workshops (ICSTW), 2014 IEEE Seventh International Conference on* (pp. 143-152). IEEE.

3. Farchi, E., Segall, I., Tzoref-Brill, R., & Zlotnick, A. (2014, March). Combinatorial Testing with Order Requirements. In *Software Testing, Verification and Validation Workshops (ICSTW), 2014 IEEE Seventh International Conference on* (pp. 118-127). IEEE.

4. Itai Segall, Rachel Tzoref-Brill and Aviad Zlotnick. Combining Minimization and Generation for Combinatorial Testing In *Software Testing, Verification and Validation Workshops (ICSTW), 2015 IEEE Sixth International Conference on* IEEE.

5. Wu, H., Nie, C., & Kuo, F. C. (2014, March). Test suite prioritization by switching cost. In *Software Testing, Verification and Validation Workshops (ICSTW), 2014 IEEE Seventh International Conference on* (pp. 133-142). IEEE.

6. Petke, J., Cohen, M., Harman, M., & Yoo, S. Practical Combinatorial Interaction Testing: Empirical Findings on Efficiency and Early Fault Detection. IEEE TSE, preprint, 2015.

7. Linbin Yu, Feng Duan, Yu Lei, Raghu N. Kacker and D. Richard Kuhn. Constraint Handling In Combinatorial Test Generation Using Forbidden Tuples

8. Mayo, Q., Michaels, R., & Bryce, R. (2014, March). Test Suite Reduction by Combinatorial-Based Coverage of Event Sequences. In *Software Testing, Verification and Validation Workshops (ICSTW), 2014 IEEE Seventh International Conference on* (pp. 128-132). IEEE.

9. Gargantini, A., & Vavassori, P. (2012, April). Citlab: a laboratory for combinatorial interaction testing. In *Software Testing, Verification and Validation (ICST), 2012 IEEE Fifth International Conference on* (pp. 559-568). IEEE.

10. Garn, B., & Simos, D. E. (2014, March). Eris: A tool for combinatorial testing of the Linux system call interface. In *Software Testing, Verification and Validation Workshops (ICSTW), 2014 IEEE Seventh International Conference on* (pp. 58-67). IEEE.

11. Arcaini, P., Gargantini, A., & Vavassori, P. (2014, March). Validation of models and tests for constrained combinatorial interaction testing. In *Software Testing, Verification and Validation Workshops (ICSTW), 2014 IEEE Seventh International Conference on* (pp. 98-107). IEEE.

12. Satish, P., Paul, A., & Rangarajan, K. (2014, March). Extracting the combinatorial test parameters and values from UML sequence diagrams. In *Software Testing, Verification and Validation Workshops (ICSTW), 2014 IEEE Seventh International Conference on* (pp. 88-97). IEEE.

13. Christopher Henard, Mike Papadakis and Yves Le Traon. Flattening or Not the Combinatorial Interaction Testing Models?, *Software Testing, Verification and Validation (ICST), 2015 IEEE Fifth International Conference*

14. Zeppetzauer, U., & Kruse, P. M. (2014, September). Automating test case design within the classification tree editor. In *Computer Science and Information Systems (FedCSIS), 2014 Federated Conference on* (pp. 1585-1590). IEEE.

15. Gargantini, A., & Vavassori, P. (2014). Efficient Combinatorial Test Generation Based on Multivalued Decision Diagrams. In *Hardware and Software: Verification and Testing* (pp. 220-235). Springer International Publishing.

16. Roberto Erick Lopez-Herrejon, Stefan Fischer, Rudolf Ramler and Alexander Egyed. A First Systematic Mapping Study on Combinatorial Interaction Testing for Software Product Lines, *Software Testing, Verification and Validation Workshops (ICSTW), 2015 IEEE Seventh International Conference on*