A Systematic Mapping Study on Non-Functional Search-Based Software Testing

Wasif Afzal, Richard Torkar and Robert Feldt
Blekinge Institute of Technology,
S-372 25 Ronneby, Sweden
{waf,rto,rfd}@bth.se

Abstract

Automated software test generation has been applied across the spectrum of test case design methods; this includes white-box (structural), black-box (functional), grey-box (combination of structural and functional) and non-functional testing. In this paper, we undertake a systematic mapping study to present a broad review of primary studies on the application of search-based optimization techniques to non-functional testing. The motivation is to identify the evidence available on the topic and to identify gaps in the application of search-based optimization techniques to different types of non-functional testing. The study is based on a comprehensive set of 35 papers obtained after using a multi-stage selection criteria and are published in workshops, conferences and journals in the time span 1996–2007. We conclude that the search-based software testing community needs to do more and broader studies on non-functional search-based software testing (NFSBST) and the results from our systematic map can help direct such efforts.

1. Introduction

Search-based software testing (SBST) research has attracted much attention in recent years as part of a general interest in search-based software engineering approaches [27, 28]. The growing interest in SBST can be attributed to the fact that there is a need for automatic generation of test data, since it is well-known that exhaustive testing is infeasible and the fact that software test data generation is considered NP-hard [36]. All approaches to SBST are based on satisfaction of a certain test adequacy criterion represented by a fitness function [27, 36]. McMinn [36] has written a survey on search-based software test generation, which shows the application of search-based techniques for white-box testing, black-box testing, grey-box testing and for the verification of non-functional properties. The survey shows that for non-functional testing, the search-based techniques are applied for execution time testing of real-time systems. Now, it is both important and interesting to know the extent of application of search-based optimization techniques for testing other non-functional properties. It is with this motivation that the current study has emerged from our work to gather, map and summarize primary studies about NFSBST in an accurate, fair and partial manner [34]. It is essentially a systematic mapping study to identify available evidence on NFSBST. A *systematic map* provides an overview of a research area to assess the quantity of evidence existing on a topic of interest [34] (see e.g. Bailey's et al. mapping study [4]).

The remainder of this paper is organized as follows. Section 2 describes our research protocol, including the search strategy and study selection. In Section 3, we describe the results. Sections 4 and 5 comprises of analysis and discussion of results, while the paper is concluded in Section 6.

2. Identification of research

We defined the following research question inline with the overall purpose of the study:

RQ: In which non-functional testing areas have search-based techniques been applied and what are the different metaheuristics used ?

A clear definition of population, intervention, outcomes and experimental design helps identifying relevant primary studies [34]. Our population is limited to the application area of software testing. Our intervention includes application of metaheuristic search techniques to test different types of non-functional properties. The outcome of our interest represents different types of non-functional testing that use metaheuristic search techniques.

2.1. Generating a search strategy

We used the following search terms to find relevant papers:

- **Population:** testing, software testing, testing software, test data generation, automated testing, automatic testing.

- **Intervention:** evolutionary, heuristic, search-based, metaheuristic, optimization, hill-climbing, simulated annealing, tabu search, genetic algorithms, genetic programming.

- **Outcomes:** non-functional, safety, robustness, stress, security, usability, integrity, efficiency, reliability, maintainability, testability, flexibility, reusability, portability, interoperability, performance, availability, scalability

We used Boolean `OR` to join alternate words and synonyms and Boolean `AND` to join major terms for population, intervention and outcome. The non-functional properties listed under outcomes are guided by five existing taxonomies, namely McCall software quality model, Boehm software quality model [19], ISO/IEC 9126-1 [30], IEEE Standard 830-1998 [29] and Donald G. Firesmith's taxonomy [20]. The non-functional properties obtained from existing taxonomies are restricted to high-level external attributes only for the sole purpose of guiding the search strategy. The different non-functional testing areas that are discussed later in the paper cannot be mapped as it is with these listed non-functional properties. Therefore, while quality of service includes attributes such as availability and reliability, we have retained the term quality of service in later part of the paper (Subsection 4.2) to remain consistent with the terms used by the original authors in their respective papers. Similarly, one can argue execution time (Subsection 4.1) and buffer overflow (Subsection 4.3) to fit under performance and security respectively, but we remain consistent with using the common terms of execution time and buffer overflow according to the authors' usage.

The search was applied on digital libraries accessed via IEEE Xplore, ACM Digital Library, Compendex and ISI Web of Science. In addition, manual search was performed on the following journals (J) and conference proceedings (C): Real Time Systems Symposium (C), Real Time Systems (J), Genetic and Evolutionary Computation Conference—Search-based Software Engineering Track (C),Software Testing, Verification and Reliability (J) and Software Quality Journal (J). To have confidence in the completeness of search, the results of the search were matched against a core set of studies to compare that the search found the entire core set.

To have a more representative set of studies, we also scanned the reference lists of primary studies and contacted researchers who authored most of the papers in a particular non-functional area. Only studies within the time span 1996–2007 were included. It is important to note that having restricted the search within these years excluded studies by Schultz et al. [42, 43] (authored in year 1992 and 1995 respectively) which applies evolutionary algorithms for robustness testing of autonomous vehicle controllers. We therefore, do not include these two studies in the analysis.

2.2. Study selection

Optimization techniques have been applied across different engineering and scientific disciplines. Moreover within software testing, search techniques have been applied from planning to execution. Therefore, it is imperative that we define comprehensive inclusion/exclusion criteria. We excluded studies that do not relate to software engineering/development, do not relate to software testing, do not report application of optimization techniques, do not report application of metaheuristics (metaheuristics include hill climbing, simulated annealing, tabu search, ant colony methods, swarm intelligence and evolutionary methods [10]), describe search-based testing approaches which are inherently structural (white-box), functional (black-box) or grey-box (combination of structural and functional) (this exclusion criterion is relaxed to include those studies where a structural test criterion is used to test non-functional properties, e.g. [5]), are not related to the testing of the end product e.g. [55], are related to test planning e.g. [16], make use of model checking and formal methods e.g. [3, 17], report performance of a particular metaheuristic instead of its application to software testing e.g. [35], report on test case prioritization e.g. [50], are used for prediction and estimation of software properties e.g. [6, 44].

In the beginning, a single researcher excluded 37 references out of a total of 404, primarily based on reading the title and abstract. The remaining 367 references were subjected to detailed exclusion criteria, which involved three researchers. This resulted in 60 remaining papers, which were further filtered out by reading full-text. A final figure of 24 primary studies was reached after excluding similar studies that were published in different venues. The 24 primary studies were complemented with 11 more papers by scanning the reference lists of the primary studies and contacting relevant authors.

3. Results

The results indicate that within non-functional testing, the application of metaheuristic search techniques can be classified under execution time, quality of service (QoS), buffer overflow, usability, and safety.

Figure 1 shows the year-wise distribution of primary studies within each non-functional property as well as the frequency of application of different metaheuristics. The

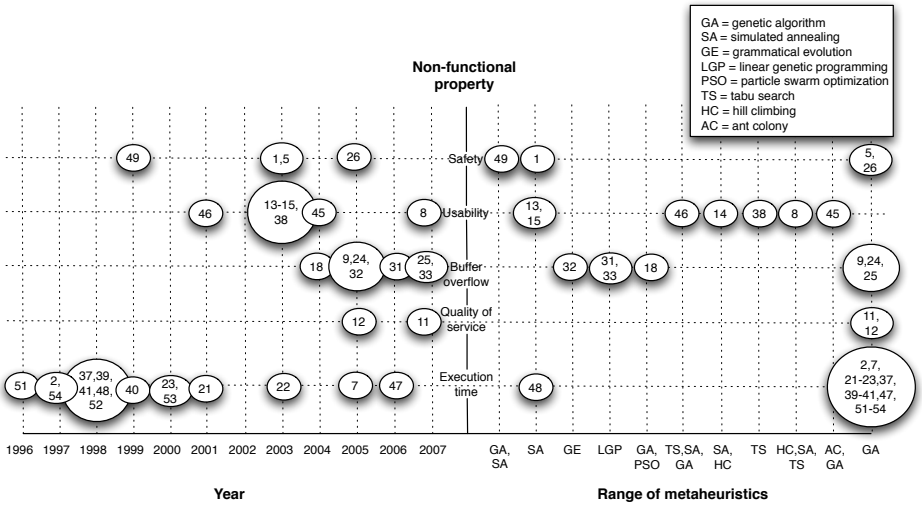

Figure 1. Distribution of NFSBST research over range of applied metaheuristics and time period.

bubble at the intersection of axes contains the reference number of papers. It is evident from the figure that genetic algorithms are the most widely used metaheuristic with applications in 21 papers across different types of non-functional testing. In the left quadrant of Figure 1, each bubble represents the reference numbers of primary studies within each non-functional area in respective years from 1996–2007.

4. Analysis

The focus of this section is to present a broad overview of research within NFSBST, discussion of range of metaheuristic techniques used and satisfaction of problem objectives.

4.1. Execution time

The application of metaheuristic search techniques to test real-time requirements in embedded computer systems involves finding the best and worst case execution times (BCET, WCET) to determine if timing constraints are fulfilled. Our systematic map indicates that the papers measuring BCET and WCET are by far the largest contributor in NFSBST research. The study by Briand et al. [7] has differentiated the temporal testing research into two directions. The one direction focuses on violation of timing constraints due to input values and has attracted the bulk of research. The other direction, which is the one taken by Briand et al. [7], analyses task architectures and consider seeding times of events triggering tasks and tasks' synchronization. This study does not considers tasks in isolation. Both approaches to temporal verification, however, are complementary. Another dimension of research into temporal testing using metaheuristic search techniques focuses on properties of test objects inhibiting evolutionary testability and formulation of complexity measures for predicting evolutionary testability [21, 22].

Genetic algorithms have been used as the metaheuristic in majority of studies (14 out of 15). The fitness functions vary according to research dimensions described above, which includes measurement of execution time of the test object, coverage of code annotations inserted along shortest and longest algorithmic execution paths and exponential fitness function based on the difference between execution's deadline and execution's actual completion.

4.2. Quality of Service (QoS)

Under the umbrella of service-oriented software engineering, genetic algorithms have been used for quality of service aware composition and violation of service level agreements (SLAs) between the integrator and the end user. The range of fitness functions used are based on the maximization of desired QoS attributes with a static or dynamic penalty function and a combination of distance-based fitness with a fitness guiding the coverage of target statements.

4.3. Buffer overflow

Buffer overflow can cause unauthorized exploits, thus compromising software security. Grammatical evolution, linear genetic programming, genetic algorithm and particle swarm optimization have been used for detecting buffer

overflows. The objective is to detect buffer overflows, vulnerable statements, exceptions and evolving plausible attacks. Most of the fitness functions are based on the ability of an attack to fulfill the conditions necessary for a successful exploit. The work of Kayacik et al. [31, 32, 33] is notable as they describe an approach to a framework for attack generation based on the evolution of system call sequences.

4.4. Usability

Search-based usability testing of software has been applied in the form of interaction testing where the goal is to test the t-way interactions taking place through the user interface. The research into interaction testing has focused on generating covering arrays which is a combinatorial object representing interactions. These studies show the use of hill-climbing, simulated annealing, tabu search, genetic algorithms and ant colony algorithms as the applied metaheuristics. The objective is either rapid coverage of interactions or obtaining smaller test suites. The fitness function used for constructing covering array is the number of uncovered t-subsets.

4.5. Safety

Search-based safety testing is an area where the research has targeted real world problems such as safety of car control systems [5] and steam boilers [1]. The research into search-based safety testing can be differentiated into two themes. One is the case where generation of separate inputs is discussed to test the safety property while the other case discusses generation of sequence of inputs. The objective is the violation of a safety property. The used metaheuristics include genetic algorithms and simulated annealing. The fitness functions used measures the cost related to the violation of the safety property.

5. Discussion

We presented the results of the initial scoping study (systematic map) to identify the extent and form of literature within NFSBST. The results of our systematic map indicates that NFSBST is focused on five areas, with execution time testing being the most researched non-functional property. This indicates that execution time testing represents a suitable search problem. On the other hand, for execution time testing this might also mark the beginning of more in-depth analysis of problem characteristics including comparative and performance evaluation studies [21, 22]. As compared to execution time testing, the application of metaheuristic search techniques for detecting buffer overflows, usability testing, safety testing and quality of service is more recent. Further feedback from empirical studies into these niche non-functional areas is required to gain confidence into the efficacy when applying search-based techniques.

We also find that the current taxonomies for non-functional properties need to assemble a more complete set of non-functional properties for software systems.

Apart from the final set of 35 papers, our search also resulted in studies which, although, applies search-based techniques, are not related to test data generation. Examples of such studies include reliability modeling [44] and test planning [16]. Studies relevant to test planning reflects the growing application of metaheuristics across the software testing lifecycle, while studies related to reliability modeling offers yet another dimension where the application of search techniques can offer near optimal solutions. These studies, together with existing SBST literature, can offer an exciting future arena where studies are not only limited to automated software test data generation but also extended to address broader verification and validation problems that are open to the application of search-based techniques. Our future work with a systematic review should explore these possibilities in more detail.

In terms of validity threats, there is a possibility that we might have missed relevant studies. However, our rigorous search strategy (Subsection 2.1) should have assembled a reasonable sample.

6. Conclusions

This work presents initial findings related to the application of metaheuristic search techniques to test non-functional properties. A total of 35 papers published in the years 1996–2007 are used a basis to map the application of metaheuristic search techniques to five different non-functional areas of execution time, quality of service, buffer overflow, usability and safety.

We presented an analysis of these studies in terms of problem objective, applied metaheuristic and range of fitness functions used. A large percentage (42.8%) of the studies deal with execution time testing with evidence of experimentation with real world applications. Regarding the rest of the non-functional properties, further feedback from empirical studies is desirable. We also found that diverse metaheuristic search techniques have been applied to achieve problem-specific objectives, with genetic algorithms being the most frequently used metaheuristic.

There is still plenty of potential for automating non-functional testing using search-based techniques. The results of our systematic map also indicate that the current body of knowledge concerning SBST does not report studies on many of the other non-functional properties. On the other hand, there is a need to extend the early optimistic results of applying NFSBST to larger real world systems, thus

moving towards a generalization of results.

Future work includes extending the presented results into a systematic literature review.

References

[1] O. Abdellatif-Kaddour, P. Thevenod-Fosse, and H. Waeselynck. Property-Oriented Testing based on Simulated Annealing. In *Proceedings of ACS/IEEE International Conference on Computer Systems and Applications (AICCSA' 2003)*, Tunis (Tunisie), 2003.

[2] J. T. Alander, T. Mantere, and G. Moghadampour. Searching Protection Relay Response Time Extremes using Genetic Algorithm – Software Quality by Optimization. In *Proceedings of the 4th International Conference on Advances in Power System Control, Operation and Management, APSCOM-97*, Hong Kong, November 1997.

[3] E. Alba and F. Chicano. Finding Safety Errors with ACO. In *Proceedings of 9th Annual Conference on Genetic and Evolutionary Computation*, ACM, New York, USA, 2007.

[4] J. Bailey, D. Budgen, M. Turner, B. Kitchenham, P. Brereton, and S. Linkman. Evidence Relating to Object-Oriented Software Design: A Survey. In *Proceedings of First International Symposium on Empirical Software Engineering and Measurement*, IEEE, 2007.

[5] A. Baresel, H. Pohlheim, and S. Sadeghipour. Structural and Functional Sequence Test of Dynamic and State-Based Software with Evolutionary Algorithms. In *Proceedings of Genetic and Evolutionary Computation Conference (GECCO 2003), Lecture Notes in Computer Science (LNCS 2724)*, Springer–Verlag, Berlin, Germany, 2003.

[6] S. Boukif, H. Sahraoui, and G. Antoniol. Simulated Annealing for Software Quality Prediction. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, ACM, New York, USA, 2006.

[7] L. C. Briand, Y. Labiche, and M. Shousha. Stress Testing Real-Time Systems with Genetic Algorithms. In *Proceedings of the 2005 Conference on Genetic and Evolutionary Computation, GECCO 05*, Washington, DC, USA, June 25–29 2005.

[8] R. C. Bryce and C. J. Colbourn. One-Test-at-a-Time Heuristic Search for Interaction Test Suits. In *Proceedings of the 2007 Conference on Genetic and Evolutionary Computation, GECCO 07*, London, UK, July 7–11, 2007.

[9] J. Budynek, E. Bonabeau, and B. Shargel. Evolving Computer Intrusion Scripts for Vulnerability Assessment and Log Analysis. In *Proceedings of the 2005 Conference on Genetic and Evolutionary Computation, GECCO 05*, pages 153–160, Washington, DC, USA, June 25–29, 2005.

[10] E. K. Burke and G. Kendall. *Search Methodologies – Introductory Tutorials in Optimization and Decision Support Techniques*. Springer Science and Business Media, New York, USA, 2005.

[11] G. Canfora, M. D. Penta, R. Esposito, and M. L. Villani. Search-based Testing of Service Level Agreements. In *Proceedings of the Conference on Genetic and Evolutionary Computation, GECCO '07*, London, UK, July 7–11, 2007.

[12] G. Canfora, M. D. Penta, R. Esposito, and M. L. Villani. An Approach for QoS-aware Service Composition based on Genetic Algorithms. In *GECCO 05: Proceedings of the 2005 Conference on Genetic and Evolutionary Computation*, Washington, DC, USA, June 25–29, 2005.

[13] M. B. Cohen, C. J. Colbourn, and A. C. H. Ling. Constructing Strength Three Covering Arrays with Augmented Annealing. *Discrete Mathematics*, 2003.

[14] M. B. Cohen, P. B. Gibbons, W. B. Mugridge, and C. J. Colbourn. Constructing Test Suites for Interaction Testing. In *Proceedings of the 25th International Conference on Software Engineering (ICSE 03)*, IEEE, 2003.

[15] M. B. Cohen, P. B. Gibbons, W. B. Mugridge, C. J. Colbourn, and J. S. Collofello. Variable Strength Interaction Testing of Components. In *Proceedings of the 27th Annual International Computer Software and Applications Conference (COMPSAC 03)*, IEEE, 2003.

[16] Y. S. Dal, M. Xie, K. L. Poh, and B. Yang. Optimal Testing-Resource Allocation with Genetic Algorithm for Modular Software Systems. *The Journal of Systems and Software*, 66(1), 2003.

[17] K. Derdarian, R. M. Hierons, M. Harman, and Q. Guo. Input Sequence Generation for Testing of Communicating Finite State Machines (CFSMs). In *Proceedings of Genetic and Evolutionary Computation Conference (GECCO 2004), Lecture Notes in Computer Science (LNCS 3103), Springer–Verlag*, Berlin, Germany, 2004.

[18] G. Dozier, D. Brown, J. Hurley, and K. Cain. Vulnerability Analysis of Immunity-Based Intrusion Detection Systems Using Evolutionary Hackers. In *Proceedings of the 2004 Conference on Genetic and Evolutionary Computation, GECCO 04*, Seattle, Washington, USA, June 26–30, 2004.

[19] N. E. Fenton and S. L. Pfleeger. *Sofware Metrics - A Rigorous and Practical Approach, 2nd Edition*. International Thomson Computer Press, Boston, USA, 1996.

[20] D. G. Firesmith. Common Concepts Underlying Safety, Security, and Survivability Engineering. Technical Note CMU/SEI-2003-TN-033, Carnegie Mellon Software Engineering Institute, 2003.

[21] H. G. Gross. A Prediction System for Dynamic Optimization-based Execution Time Analysis. In *Proceedings of First International Workshop on Software Engineering using Metaheuristic Innovative Algorithms (SEMINAL), ICSE 2001*, Toronto, 2001.

[22] H. G. Gross. An Evaluation of Dynamic, Optimization-based Worst-case Execution Time Analysis. In *Proceedings of the International Conference on Information Technology: Prospects and Challenges in the 21st Century*, Kathmandu, Nepal, 2003.

[23] H. G. Gross, B. F. Jones, and D. E. Eyres. Structural Performance Measure of Evolutionary Testing Applied to Worst-case Timing of Real-time Systems. *Proceedings of IEE Software*, 147(2), 2000.

[24] C. Grosso, G. Antoniol, M. D. Penta, P. Galinier, and E. Merlo. Improving Network Applications Security: a New Heuristic to Generate Stress Testing Data. In *Proceedings of the Annual Conference on Genetic and Evolutionary Computation, GECCO 05*, ACM, New York, USA, 2005.

[25] C. D. Grosso, G. Antoniol, E. Merlo, and P. Galinear. Detecting Buffer Overow via Automatic Test Input Data Generation. *Computers and Operations Research, Elsevier*, 2007.

[26] A. G. H. Pohlheim, M. Conrad. Evolutionary Safety Testing of Embedded Control Software by Automatically Generating Compact Test Data Sequences. In *SAE 2005 World Congress Exhibition*, Detroit, MI, USA, April 2005.

[27] M. Harman. The Current State and Future of Search-based Software Engineering. In *Proceedings of Future of Software Engineering (FOSE 07) at 29th International Conference on Software Engineering*. IEEE Computer Society, USA, 2007.

[28] M. Harman and B. Jones. Search-based Software Engineering. *Information and Software Technology*, 43(14), 2001.

[29] IEEE Std 830-1998. *IEEE Recommended Practice for Software Requirements Specifications*, 1998.

[30] International Standard. *ISO/IEC 9126-1:2001 Software Engineering Product Quality Part 1: Quality Model*, 2001.

[31] H. G. Kayacik, M. Heywood, and A. N. Zincir-Heywood. On Evolving Buffer Overflow Attacks Using Genetic Programming. In *Proceedings of the 2006 Conference on Genetic and Evolutionary Computation, GECCO 06*, Seattle, Washington, USA, July 8–12, 2006.

[32] H. G. Kayacik, A. N. Zincir-Heywood, and M. Heywood. Evolving Successful Stack Overflow Attacks for Vulnerability Testing. In *Proceedings of the 21st Annual Computer Security Applications Conference (ACSAC 2005)*, IEEE, 2005.

[33] H. G. Kayacik, A. N. Zincir-Heywood, and M. Heywood. Automatically Evading IDS Using GP Authored Attacks. In *Proceedings of the IEEE Computational intelligence in Security and Defense Applications - CISDA 2007*, pages 153–160, April 2007.

[34] B. Kitchenham. Guidelines for Performing Systematic Literature Reviews in Software Engineering. Technical Report EBSE-2007-01, Keele University and University of Durham, UK, 2007.

[35] J. Koljonen, M. Mannila, and M. Wanne. Testing the Performance of a 2D Nearest Point Algorithm with Genetic Algorithm Generated Gaussian Distributions. *Expert Systems with Applications*, 32(3), 2007.

[36] P. McMinn. Search-Based Software Test Data Generation: A Survey. *Software Testing,Verification and Reliability*, 14(2), 2004.

[37] F. Mueller and J. Wegener. A Comparison of Static Analysis and Evolutionary Testing for the Verication of Timing Constraints. In *Proceedings of the 4th IEEE Real-Time Technology and Applications Symposium*, Denver, USA, June 1998.

[38] K. J. Nurmela. Upper Bounds for Covering Arrays by Tabu Search. *Discrete Applied Mathematics, Elsevier*, 2003.

[39] M. OSullivan, S. Vossner, and J. Wegener. Testing Temporal Correctness of Real-Time Systems - A New Approach using Genetic Algorithms and Cluster Analysis. In *Proceedings of the 6th European Conference on Software Testing, Analysis Review (EuroSTAR 1998)*, Munich, Germany, 1998.

[40] H. Pohlheim and J. Wegener. Testing the Temporal Behavior of Real-Time Software Modules using Extended Evolutionary Algorithms. In *Proceedings of Genetic and Evolutionary Computation Conference (GECCO)*, 1999.

[41] P. Puschner and R. Nossal. Testing the Results of Static Worst-Case Execution-Time Analysis. In *Proceedings of the 19th IEEE Real-Time Systems Symposium (RTSS '98)*, Madrid, Spain, December 1998.

[42] A. C. Schultz, J. J. Grefenstette, and K. A. D. Jong. Adaptive Testing of Controllers for Autonomous Vehicles. In *Proceedings of the 1992 Symposium on Autonomous Underwater Vehicle Technology*. IEEE, 1992.

[43] A. C. Schultz, J. J. Grefenstette, and K. A. D. Jong. Learning to Break Things: Adaptive Testing of Intelligent Controllers. Naval Research Laboratory, Oxford University Press, 1995.

[44] A. Sheta. Reliability Growth Modeling for Software Fault Detection Using Particle Swarm Optimization. In *IEEE Congress on Evolutionary Computation*. IEEE, 2006.

[45] T. Shiba, T. Tsuchiya, and T. Kikuno. Using Artificial Life Techniques to Generate Test Cases for Combinatorial Testing. In *Proceedings of the 28th Annual International Computer Software and Applications Conference (COMPSAC 04)*, IEEE, 2004.

[46] J. Stardom. Metaheuristics and the Search for Covering and Packing Arrays. Masters Thesis, Simon Fraser University, 2001.

[47] M. Tlili, S. Wappler, and H. Sthamer. Improving Evolutionary Real-Time Testing. In *Proceedings of the 2006 Conference on Genetic and Evolutionary Computation, GECCO 06*, Seattle, Washington, USA, July 8–12 2006.

[48] N. Tracey, J. Clark, and K. Mander. The Way Forward for Unifying Dynamic Test Case Generation: The Optimisation – Based Approach. In *International Workshop on Dependable Computing and Its Applications, IFIP*, 1998.

[49] N. J. Tracey, J. Clark, J. McDermid, and K.Mander. Integrating Safety Analysis with Automatic Test Data Generation for Software Safety Verification. In *Proceedings of the 17th International Conference on System Safety*, IEEE, August 1999.

[50] K. R. Walcott, M. Soffa, G. M. Kapfhammer, and R. Roos. TimeAware Test Suite Prioritization. In *Proceedings of the 2006 International Symposium on Software Testing and Analysis*, ACM, New York, USA, 2006.

[51] J. Wegener, K. Grimm, M. Grochtmann, H. Sthamer, and B. Jones. Systematic Testing of Real-time Systems. In *Proceedings of the 4th European Conference on Software Testing, Analysis Review (EuroSTAR 1996)*, Amsterdam, Netherlands, December 1996.

[52] J. Wegener and M. Grochtmann. Verifying Timing Constraints of Real-Time Systems by Means of Evolutionary Testing. *Real-time Systems*, 1998.

[53] J. Wegener, R. Pitschinetz, and H. Sthamer. Automated Testing of Real-Time Tasks. In *Proceedings of the 1st International Workshop on Automated Program Analysis, Testing and Verification*, Limerick, Ireland, June 2000.

[54] J. Wegener, H. Sthamer, B. F. Jones, and D. E. Eyres. Testing Real-time Systems Using Genetic Algorithms. *Software Quality Journal*, 6(2):127–135, June 1997.

[55] Y. Zhan and J. Clark. Search-based Automatic Test-Data Generation at an Architectural Level. In *Proceedings of Genetic and Evolutionary Computation Conference (GECCO 2004), Lecture Notes in Computer Science (LNCS 3103)*, Springer–Verlag, Berlin, Germany, 2004.