Why the Virtual Nature of Software Makes it Ideal for Search Based Optimization

Mark Harman

CREST, King's College London, Strand, London,
WC2R 2LS, United Kingdom.

Abstract. This paper[1] provides a motivation for the application of search based optimization to Software Engineering, an area that has come to be known as Search Based Software Engineering (SBSE). SBSE techniques have already been applied to many problems throughout the Software Engineering lifecycle, with new application domains emerging on a regular basis. The approach is very generic and therefore finds wide application in Software Engineering. It facilitates automated and semi-automated solutions in situations typified by large complex problem spaces with multiple competing and conflicting objectives. Previous work has already discussed, in some detail, the advantages of the SBSE approach for Software Engineering. This paper summarises previous work and goes further, by arguing that Software Engineering provides *the ideal* set of application problems for which optimization algorithms are supremely well suited.

Key words: SBSE, Search Based Optimization, Search Based Testing, Metaheuristic Search, Optimization Algorithms

1 Introduction

We often speak of 'Software Engineering' without thinking too deeply about what it means to have a discipline of 'engineering' that considers the primary material to be 'software'. By considering both the 'engineering' aspects of 'Software Engineering' and also the unique properties of 'software' as an engineering material, this paper makes an argument that search based optimization techniques are ideally suited to Software Engineering.

That is, though all other engineering disciplines have also provided rich sources of application for search based optimization, it is in its application to problems in Software Engineering that these techniques can find greatest application. This acts as a secondary motivation for the field of SBSE. The primary

[1] This paper is written to accompany the author's keynote presentation at Fundamental Approaches to Software Engineering (FASE 2010). The talk provides an overview of SBSE and its applications and motivation. The paper focusses on the argument that the virtual nature of software makes it ideal for SBSE, since other aspects of SBSE mentioned in the FASE keynote have been covered by the author's previous keynotes and invited papers.

motivation for SBSE comes from the simple observation that these techniques do, indeed, apply well in other engineering disciplines and that, therefore, should we wish to regard Software Engineering as truly an engineering discipline, then it would only be natural to consider the application of search based optimization techniques. This form of advocacy for SBSE has been put forward by this and other authors before [CDH+03, HJ01, Har07b, Har07a, Har07c].

The acceptance of SBSE as a well-defined and worthwhile activity within the rich and diverse tapestry of Software Engineering is reflected by the increasing number of survey papers on SBSE [ABHPWar, ATF09, HMZ09, McM04, Räi09]. Further evidence for widespread interest and uptake, comes from the many special issues, workshops and conferences on the topic. However, this paper seeks to go a step further. It argues that Software Engineering is not merely an acceptable subject for the application of search based optimization, but that it is even better suited than all other areas of engineering activity, as a result of the very special properties of software as an engineering material.

2 Overview of SBSE

The existing case for SBSE in the literature rests upon the observation that

> "Software engineers often face problems associated with the balancing of competing constraints, trade-offs between concerns and requirement imprecision. Perfect solutions are often either impossible or impractical and the nature of the problems often makes the definition of analytical algorithms problematic." [HJ01]

The term SBSE was first used by Harman and Jones [HJ01] in 2001. The term 'search' is used to refer to the metaheuristic search–based optimization techniques. Search Based Software Engineering seeks a fundamental shift of emphasis from solution construction to solution description. Rather than devoting human effort to the task of finding solutions, the search for solutions is *automated* as a search, guided by a fitness function, defined by the engineer to capture *what* is required rather than *how* it is to be constructed. In many ways, this approach to Software Engineering echoes, at the macro level of Software Engineering artifacts, the declarative programming approach [DB77], which applies at the code level; both seek to move attention from the question of 'how' a solution is to be achieved to the question of 'what' properties are desirable.

Harman and Jones argued that SBSE could become a coherent field of activity that combines the expertise and skills of the Metaheuristic Search community with those of the Software Engineering community. Though there was previous work on the application of search based optimization to Software Engineering problems [CCHA94, JES98, TCM98, XES+92], the 2001 paper was the first to articulate SBSE as a field of study in its own right and to make a case for its wider study.

Since the 2001 paper, there has been an explosion of SBSE activity, with evidence for a rapid increase in publications on the topic [HMZ09]. For example, SBSE has been applied to testing [BSS02, Bot02, BLS05, GHHD05, HHH$^+$04, MHBT06, WBS01], bug fixing [AY08, WNGF09] design, [HHP02, MM06, SBBP05], requirements, [BRSW01, ZFH08], project management [AC07, ADH04, KSH02] and refactoring. [OÓ06, HT07]. There have been SBSE special issues in the journals Information and Software Technology (IST), Software Maintenance and Evolution (JSME) and Computers and Operations Research (COR) with forthcoming special issues in Empirical Software Engineering (EMSE), Software Practice and Experience (SPE), Information and Software Technology (IST) and IEEE Transactions on Software Engineering (TSE). There is also an established Symposium on Search Based Software Engineering (SSBSE), a workshop on Search Based Software Testing (SBST) and a dedicated track of the Genetic and Evolutionary Computation COnference (GECCO) on SBSE.

2.1 All you need is love *of optimization*; you already have Representation and Fitness

Getting initial results from search based algorithms applied to Software Engineering is relatively straightforward. This has made SBSE attractive to researchers and practitioners from the Software Engineering community. Becoming productive as a Search Based Software Engineer does not required a steep learning curve, nor years of apprenticeship in the techniques, foundations and nomenclature of Optimization Algorithms. It has been stated [Har07d, HJ01] that there are only two key ingredients required:

1. The choice of the representation of the problem.
2. The definition of the fitness function.

Of course, a Software Engineer is very likely to have, already at their disposal, a workable representation for their problem. Furthermore, Harman and Clark argue that

> "Metrics are Fitness functions too" [HC04].

They argue that the extensive body of literature on metrics and software measurement can be mined for candidate fitness functions. This would allow Software Engineers to optimize according to software measurements, rather than merely to passively measure software artifacts. Though every metric may not be effective, because some may fail to measure what they claim to measure [She95], this need not be a problem. Indeed, one of the attractive aspects of metrics as fitness functions, is that such failings on the part of the metrics will become immediately evident through optimization. Harman and Clark show that there is a close connection between metrics as fitness functions and empirical assessment of the representation condition of software measurement.

2.2 Algorithms

The most widely used algorithms in SBSE work have, hitherto [HMZ09], been local search, simulated annealing genetic algorithms and genetic programming. However, other authors have experimented with other search based optimizers such as parallel EAs [AC08], evolution strategies [AC05], Estimation of Distribution Algorithms (EDAs) [SL08], Scatter Search [BTDD07, AVCTV06, Sag07], Particle Swarm Optimization (PSO) [LI08, WWW07], Tabu Search [DTBD08] and Local search [KHC$^+$05].

3 Why Software Engineering is the ‘Killer Application’ for search based optimization

Previous work has considered the motivation for SBSE in terms of the advantages it offers to the Software Engineer. For instance it has been argued [HMZ09, Har07b] that SBSE is

1. **Scalable**, because of the ‘embarrassingly parallel’ [Fos95] nature of the underlying algorithms which can yield orders of magnitude scale up over sequential implementations [LB08].
2. **Generic**, due to the wide prevalence of suitable representations and fitness functions, right across the Software Engineering spectrum.
3. **Robust**, due to the ability of search based optimization to cope with noise, partial data and inaccurate fitness.
4. **Insight-rich**, as a result of the way in which the search process itself can shed light on the problems faced by decision makers.
5. **Realistic**, due to the way in which SBSE caters naturally for multiple competing and conflicting engineering objectives.

These five features of SBSE are important and have been described in more detail elsewhere [HMZ09, Har07b]. However, most are reasons for the use of search based optimization in general. They apply equally well to any class of optimization problems, both within and without the field of Software Engineering. This does not make them any less applicable to Software Engineering. However, it does raise the question as to whether there are any special software-centric reasons why SBSE should be considered to be an attractive, important and valuable field of study in its own right. That is, we ask:

> Are there features of Software Engineering problems that make search based optimization particularly attractive?

Perhaps unsurprisingly, the author’s answer to this question is: ’yes’. The rest of this paper seeks to explain why.

In more traditional engineering disciplines, such as mechanical, biomedical, chemical, electrical and electronic engineering, search based optimization has been applied for many years [BW96, CHS98, LT92, PCV95]. These applications denote a wide spectrum of engineering activity, from well-established traditional fields of engineering to more recent innovations. However, for each, it has been possible and desirable, to optimize using search based optimization. This is hardly surprising. After all, surely engineering is all about optimization. When we speak of finding an engineering solution, do we not include balancing competing practical objectives in the best way possible. It should not be surprising, therefore, that optimization algorithms play a very important role.

In all of these fields of engineering, the application of optimization techniques provides the engineer with a mechanism to consider many candidate solutions, searching for those that yield an acceptable balance of objectives. The advent of automatic high speed computation in the past sixty years has provided a huge stimulus to the optimization community; it has allowed this search to be automated. Guided by a fitness function, automated search is one of the most profitable and archetypal applications of computation. It allows a designer to focus on the desired properties of a design, without having to care about implementation details.

It is the advent of software and the platforms on which it executes that has facilitated enormous breakthroughs in optimization methods and techniques. However, it is only comparatively recently that Software Engineering has started to catch up with this trend within the wider engineering community. This seems curious, since search based optimization can be viewed as a software technology. Perhaps it reflects the comparatively recent realization that the activity of designing and building software-based systems is, indeed, an engineering activity and thus one for which an optimization-based world view is important.

When we speak of software we mean more than merely the code. We typically include requirements, designs, documentation and test cases. We also include the supporting logical infrastructure of configuration control, development environments, test harnesses, bug tracking, archives and other virtual information-based resources that form part of the overall system and its development history. The important unifying property of all of this information is that it is purely logical and without any physical manifestation. As every software engineer knows, software is different to any and every other engineering artifact; very different. One cannot see, hear, smell, touch nor taste it because it has no physical manifestation.

This apparently trite observation is *so* obvious that its importance can sometimes be overlooked, for it is precisely this *virtual* nature of software makes it even better suited to search based optimization than traditional engineering artifacts. The materials with which we perform the automated search are made of the same 'virtual stuff' as the artifacts we seek to optimize. This has profound implications for the conduct of search based optimization because it directly impacts the two key ingredients of representation and fitness (see Figure 1).

In traditional engineering optimization, the artifact to be optimized is often simulated. This is typically necessary precisely because the artifact to be optimized is a physical entity. For instance, if one wants to optimize and aircraft

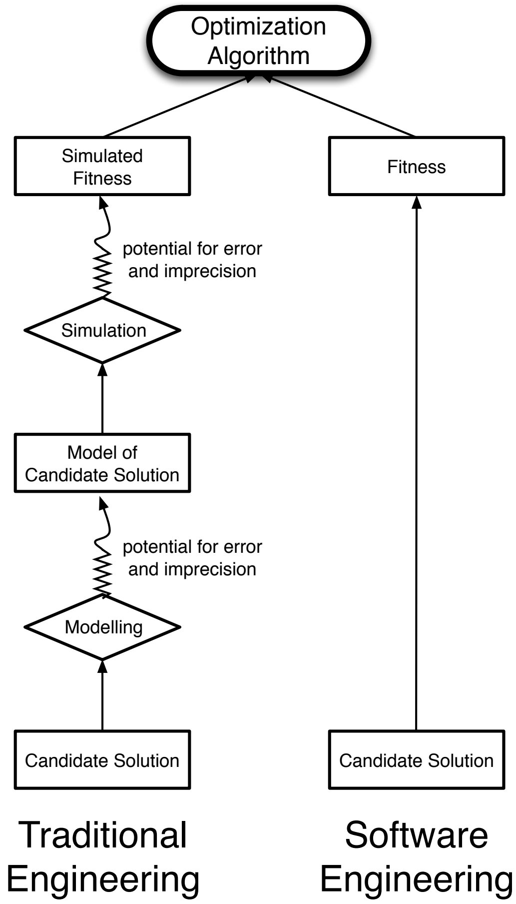

Fig. 1. Direct Application of Optimization is Possible with SBSE

engine, one cannot search the space of real engines; building even a small subset of such candidate engine designs would be prohibitively expensive. Rather, one builds of *model* of the engine (in *software*), capturing, hopefully realistically and correctly, those aspects of performance that are of interest. Furthermore, in order to compute fitness some form of simulation of the model is required. This allows us to explore the space of possible engine models, guided by a simulation of their likely real fitness.

Modelling and simulation create two layers of indirection and consequent potential for error. The model may not be entirely accurate. Indeed, if we are able to build a perfect model, then perhaps we would know so much about the engineering problem that we would be less likely to need to employ optimization. The fitness of each candidate model considered is calculated indirectly, in terms of the performance of the model with respect to some simulation of its real world behaviour. Once again, this introduces indirection and with it, the potential for error, imprecision and misunderstanding.

Contrast this traditional, physical engineering scenario with that of SBSE. For instance, consider the widely studied problem of finding test data (for example, to traverse a chosen branch of interest [ABHPWar, HMZ09, McM04]). For this problem there is no need for a model of the software to be tested nor the test case to be applied. Rather than modelling the test case, the optimization is applied directly to a vector which *is* the input to the program under test. Furthermore, in order to compute fitness, one need not *simulate* the execution, one may simply execute directly.

Of course, some instrumentation is required to facilitate fitness assessment. This can create issues for measurement if, for example, non–functional properties are to be optimized [ATF09, HMZ09]. These bare a superficial similarity to those present with simulations. The instrumented program is not the real program; it could be thought of as a kind of model. However, the instrumented program is clearly much closer to the original program under test than a simulation of an engine is to a real physical engine.

Furthermore, many software testing objectives, such as the structural test adequacy criteria [ABHPWar, HMZ09, McM04] are entirely unaffected by instrumentation and so there is no indirection at all. This observation applies in may aspects of software engineering. The problem of finding suitable sets of requirements operates on the requirements sets themselves. This is also true for optimization of regression test sets and for optimization of project plans and architectures.

Of course, there are some aspects of software systems which are modelled. Indeed, there is an increasing interest in model driven development. When SBSE is applied to these models, at the design level [Räi09], it may be the case that search based optimization for Software Engineering acquires a closer similarity to search based optimization for Traditional Engineering. Nevertheless, there will remain many applications for which SBSE is ideally suited to the problem because the engineering artifact is optimized directly (not in terms of a model) and the fitness is computed directly from the artifact itself (not from a simulation thereof).

4 Conclusions

Search Based Software Engineering (SBSE) is a newly emergent paradigm for both Software Engineering community and the Metaheuristic Search and optimization communities. SBSE has had notable successes and there is an increasingly widespread application of SBSE across the full spectrum of Software Engineering activities and problems. This paper is essentially a 'position paper' that argues that the unique 'virtual' property of software as an engineering material makes it ideally suited among engineering materials for search based optimization. Software Engineers can build candidate Software Engineering artifacts with comparative ease and little cost compared to traditional engineers, faced with physical artifact construction and consequent cost. In general, the Software Engineer can also measure fitness directly, not in terms of a (possibly imprecise or misrepresented) simulation of real world operation.

5 Author Biography

Mark Harman is professor of Software Engineering in the Department of Computer Science at King's College London. He is widely known for work on source code analysis and testing and he was instrumental in the founding of the field of Search Based Software Engineering, the topic of this keynote. He has given 14 keynote invited talks on SBSE and its applications in the past four years. Professor Harman is the author of over 150 refereed publications, on the editorial board of 7 international journals and has served on 90 programme committees. He is director of the CREST centre at King's College London.

6 Acknowledgements

The author is grateful for ongoing discussion and collaboration with many other colleagues in the SBSE community, sadly too numerous to list by name here. His work is currently funded by the EPSRC project Software Engineering by Automated Search (EP/D050863, SEBASE, 2006-2011), for which the other principal investigators are John Clark (University of York) and Xin Yao (University of Birmingham) and by an EPSRC platform grant (EP/G060525, CREST, 2009-2014). He was also supported by the recently completed EU Specific Targeted Research Project: Evolutionary Testing (IST-33472, EvoTest, 2006-2009). This paper draws on work done by the author on these projects.

Bibliography

[ABHPWar] Shaukat Ali, Lionel C. Briand, Hadi Hemmati, and Rajwinder Kaur Panesar-Walawege. A systematic review of the application and empirical investigation of search-based test-case generation. *IEEE Transactions on Software Engineering*, To appear.

[AC05] Enrique Alba and Francisco Chicano. Software Testing with Evolutionary Strategies. In *Proceedings of the 2nd Workshop on Rapid Integration of Software Engineering Techniques (RISE '05)*, volume 3943 of *Lecture Notes in Computer Science*, pages 50–65, Heraklion, Crete, Greece, 8-9 September 2005. Springer.

[AC07] Enrique Alba and Francisco Chicano. Software Project Management with GAs. *Information Sciences*, 177(11):2380–2401, June 2007.

[AC08] Enrique Alba and Francisco Chicano. Observations in using Parallel and Sequential Evolutionary Algorithms for Automatic Software Testing. *Computers & Operations Research*, 35(10):3161–3183, October 2008.

[ADH04] Giulio Antoniol, Massimiliano Di Penta, and Mark Harman. Search-based Techniques for Optimizing Software Project Resource Allocation. In *Proceedings of the 2004 Conference on Genetic and Evolutionary Computation (GECCO '04)*, volume 3103/2004 of *Lecture Notes in Computer Science*, pages 1425–1426, Seattle, Washington, USA, 26-30 June 2004. Springer Berlin / Heidelberg.

[ATF09] Wasif Afzal, Richard Torkar, and Robert Feldt. A systematic review of search-based testing for non-functional system properties. *Information and Software Technology*, 51(6):957–976, 2009.

[AVCTV06] Ramón Alvarez-Valdes, E. Crespo, José Manuel Tamarit, and F. Villa. A Scatter Search Algorithm for Project Scheduling under Partially Renewable Resources. *Journal of Heuristics*, 12(1-2):95–113, March 2006.

[AY08] Andrea Arcuri and Xin Yao. A Novel Co-evolutionary Approach to Automatic Software Bug Fixing. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '08)*, pages 162–168, Hongkong, China, 1-6 June 2008. IEEE Computer Society.

[BLS05] Lionel C. Briand, Yvan Labiche, and Marwa Shousha. Stress Testing Real-Time Systems with Genetic Algorithms. In *Proceedings of the 2005 Conference on Genetic and Evolutionary Computation (GECCO '05)*, pages 1021–1028, Washington, D.C., USA, 25-29 June 2005. ACM.

[Bot02] Leonardo Bottaci. Instrumenting Programs with Flag Variables for Test Data Search by Genetic Algorithms. In *Proceedings of the 2002 Conference on Genetic and Evolutionary Computation*

(GECCO '02), pages 1337–1342, New York, USA, 9-13 July 2002. Morgan Kaufmann Publishers.

[BRSW01] A. J. Bagnall, V. J. Rayward-Smith, and I. M. Whittley. The Next Release Problem. *Information and Software Technology*, 43(14):883–890, December 2001.

[BSS02] André Baresel, Harmen Sthamer, and Michael Schmidt. Fitness Function Design to Improve Evolutionary Structural Testing. In *Proceedings of the 2002 Conference on Genetic and Evolutionary Computation (GECCO '02)*, pages 1329–1336, New York, USA, 9-13 July 2002. Morgan Kaufmann Publishers.

[BTDD07] Raquel Blanco, Javier Tuya, Eugenia Daz, and B. Adenso Daz. A Scatter Search Approach for Automated Branch Coverage in Software Testing. *International Journal of Engineering Intelligent Systems (EIS)*, 15(3):135–142, September 2007.

[BW96] Peter J. Bentley and Jonathan P. Wakefield. Generic representation of solid geometry for genetic search. *Microcomputers in Civil Engineering*, 11(3):153–161, 1996.

[CCHA94] Carl K. Chang, Chikuang Chao, Su-Yin Hsieh, and Yahya Alsalqan. SPMNet: a Formal Methodology for Software Management. In *Proceedings of the 18th Annual International Computer Software and Applications Conference (COMPSAC '94)*, pages 57–57, Taipei, Taiwan, 9-11 November 1994. IEEE.

[CDH+03] John Clark, José Javier Dolado, Mark Harman, Robert Mark Hierons, Bryan Jones, Mary Lumkin, Brian Mitchell, Spiros Mancoridis, Kearton Rees, Marc Roper, and Martin Shepperd. Reformulating software engineering as a search problem. *IEE Proceedings — Software*, 150(3):161–175, 2003.

[CHS98] O. Cordón, F. Herrera, and L. Sánchez. Evolutionary learning processes for data analysis in electrical engineering applications. In D. Quagliarella, J. Périaux, C. Poloni, and G. Winter, editors, *Genetic Algorithms and Evolution Strategy in Engineering and Computer Science*, pages 205–224. John Wiley and Sons, Chichester, 1998.

[DB77] John Darlington and Rod M. Burstall. A tranformation system for developing recursive programs. *Journal of the ACM*, 24(1):44–67, 1977.

[DTBD08] Eugenia Díaz, Javier Tuya, Raquel Blanco, and José Javier Dolado. A Tabu Search Algorithm for Structural Software Testing. *Computers & Operations Research*, 35(10):3052–3072, October 2008.

[Fos95] Ian Foster. *Designing and building parallel programs:Concepts and tools for parallel software*. Addison-Wesley, 1995.

[GHHD05] Qiang Guo, Robert Mark Hierons, Mark Harman, and Karnig Derderian. Constructing Multiple Unique Input/Output Sequences using Evolutionary Optimisation Techniques. *IEE Proceedings - Software*, 152(3):127–140, June 2005.

[Har07a] Mark Harman. Automated test data generation using search based software engineering (keynote). In *2nd Workshop on Automation of Software Test (AST 07) at the 29th International Conference on Software Engineering (ICSE 2007)*, Minneapolis, USA, 2007.

[Har07b] Mark Harman. The current state and future of search based software engineering. In Lionel Briand and Alexander Wolf, editors, *Future of Software Engineering 2007*, pages 342–357, Los Alamitos, California, USA, 2007. IEEE Computer Society Press.

[Har07c] Mark Harman. Search based software engineering for program comprehension (keynote). In *15th International Conference on Program Comprehension (ICPC 2007)*, Banff, Canada, 2007.

[Har07d] Mark Harman. The Current State and Future of Search Based Software Engineering. In Lionel Briand and Alexander Wolf, editors, *Proceedings of International Conference on Software Engineering / Future of Software Engineering 2007 (ICSE/FOSE '07)*, pages 342–357, Minneapolis, Minnesota, USA, 20-26 May 2007. IEEE Computer Society.

[HC04] Mark Harman and John A. Clark. Metrics Are Fitness Functions Too. In *Proceedings of the 10th IEEE International Symposium on Software Metrics (METRICS '04)*, pages 58–69, Chicago, USA, 11-17 September 2004. IEEE Computer Society.

[HHH+04] Mark Harman, Lin Hu, Robert M. Hierons, Joachim Wegener, Harmen Sthamer, André Baresel, and Marc Roper. Testability Transformation. *IEEE Transaction on Software Engineering*, 30(1):3–16, January 2004.

[HHP02] Mark Harman, Robert Hierons, and Mark Proctor. A New Representation and Crossover Operator for Search-based Optimization of Software Modularization. In *Proceedings of the 2002 Conference on Genetic and Evolutionary Computation (GECCO '02)*, pages 1351–1358, New York, USA, 9-13 July 2002. Morgan Kaufmann Publishers.

[HJ01] Mark Harman and Bryan F. Jones. Search-based Software Engineering. *Information & Software Technology*, 43(14):833–839, December 2001.

[HMZ09] Mark Harman, Afshin Mansouri, and Yuanyuan Zhang. Search based software engineering: A comprehensive analysis and review of trends techniques and applications. Technical Report TR-09-03, Department of Computer Science, King's College London, April 2009.

[HT07] Mark Harman and Laurence Tratt. Pareto Optimal Search Based Refactoring at the Design Level. In *Proceedings of the 9th annual Conference on Genetic and Evolutionary Computation (GECCO '07)*, pages 1106–1113, London, England, 7-11 July 2007. ACM.

[JES98] Bryan F. Jones, David E. Eyres, and Harmen-H. Sthamer. A Strategy for using Genetic Algorithms to Automate Branch and Fault-based Testing. *Computer Journal*, 41(2):98–107, 1998.

[KHC+05] Bogdan Korel, Mark Harman, S. Chung, P. Apirukvorapinit, R. Gupta, and Q. Zhang. Data Dependence Based Testability Transformation in Automated Test Generation. In *Proceedings of the 16th IEEE International Symposium on Software Reliability Engineering (ISSRE '05)*, pages 245–254, Chicago, Illinios, USA, November 2005. IEEE Computer Society.

[KSH02] Colin Kirsopp, Martin Shepperd, and John Hart. Search Heuristics, Case-based Reasoning And Software Project Effort Prediction. In *Proceedings of the 2002 Conference on Genetic and Evolutionary Computation (GECCO '02)*, pages 1367–1374, New York, 9-13 July 2002. Morgan Kaufmann Publishers.

[LB08] William B. Langdon and Wolfgang Banzhaf. A SIMD interpreter for genetic programming on GPU graphics cards. In *Proceedings of the 11th European Conference on Genetic Programming, EuroGP 2008*, volume 4971 of *Lecture Notes in Computer Science*, pages 73–85, Naples, 26-28 March 2008. Springer.

[LI08] Raluca Lefticaru and Florentin Ipate. Functional Search-based Testing from State Machines. In *Proceedings of the First International Conference on Software Testing, Verfication and Validation (ICST 2008)*, pages 525–528, Lillehammer, Norway, 9-11 April 2008. IEEE Computer Society.

[LT92] J. E. Labossiere and N. Turrkan. On the optimization of the tensor polynomial failure theory with a genetic algorithm. *Transactions of the Canadian Society for Mechanical Engineering*, 16(3-4):251–265, 1992.

[McM04] Phil McMinn. Search-based Software Test Data Generation: A Survey. *Software Testing, Verification and Reliability*, 14(2):105–156, 2004.

[MHBT06] Phil McMinn, Mark Harman, David Binkley, and Paolo Tonella. The Species per Path Approach to Search-based Test Data Generation. In *Proceedings of the 2006 International Symposium on Software Testing and Analysis (ISSTA '06)*, pages 13–24, Portland, Maine, USA., 17-20 July 2006. ACM.

[MM06] Brian S. Mitchell and Spiros Mancoridis. On the Automatic Modularization of Software Systems using the Bunch Tool. *IEEE Transactions on Software Engineering*, 32(3):193–208, March 2006.

[OÓ06] Mark O'Keeffe and Mel Ó Cinnéide. Search-based Software Maintenance. In *Proceedings of the Conference on Software Maintenance and Reengineering (CSMR '06)*, pages 249–260, Bari, Italy, 22-24 March 2006. IEEE Computer Society.

[PCV95] R. Poli, S. Cagnoni, and G. Valli. Genetic design of optimum linear and nonlinear QRS detectors. *IEEE Transactions on Biomedical Engineering*, 42(11):1137–41, November 1995.

[Räi09] Outi Räihä. A Survey on Search-Based Software Design. Technical Report D-2009-1, Department of Computer Sciences University of Tampere, March 2009.

[Sag07] Ramón Sagarna. *An Optimization Approach for Software Test Data Generation: Applications of Estimation of Distribution Algorithms and Scatter Search*. PhD thesis, University of the Basque Country, San Sebastian, Spain, January 2007.

[SBBP05] Olaf Seng, Markus Bauer, Matthias Biehl, and Gert Pache. Search-based Improvement of Subsystem Decompositions. In *Proceedings of the 2005 Conference on Genetic and Evolutionary Computation (GECCO '05)*, pages 1045–1051, Washington, D.C., USA, 25-29 June 2005. ACM.

[She95] Martin J. Shepperd. *Foundations of software measurement*. Prentice Hall, 1995.

[SL08] Ramón Sagarna and José A. Lozano. Dynamic Search Space Transformations for Software Test Data Generation. *Computational Intelligence*, 24(1):23–61, February 2008.

[TCM98] Nigel Tracey, John Clark, and Keith Mander. The Way Forward for Unifying Dynamic Test-Case Generation: the Optimisation-based Approach. In *Proceedings of the IFIP International Workshop on Dependable Computing and Its Applications (DCIA '98)*, pages 169–180, Johannesburg, South Africa, 12-14 January 1998. University of the Witwatersrand.

[WBS01] Joachim Wegener, André Baresel, and Harmen Sthamer. Evolutionary Test Environment for Automatic Structural Testing. *Information and Software Technology Special Issue on Software Engineering using Metaheuristic Innovative Algorithms*, 43(14):841–854, December 2001.

[WNGF09] Westley Weimer, Thanh Vu Nguyen, Claire Le Goues, and Stephanie Forrest. Automatically finding patches using genetic programming. In *International Conference on Software Engineerign (ICSE 2009)*, pages 364–374, Vancouver, Canada, 2009.

[WWW07] Andreas Windisch, Stefan Wappler, and Joachim Wegener. Applying Particle Swarm Optimization to Software Testing. In *Proceedings of the 9th annual Conference on Genetic and Evolutionary Computation (GECCO '07)*, pages 1121–1128, London, England, 7-11 July 2007. ACM.

[XES+92] S. Xanthakis, C. Ellis, C. Skourlas, A. Le Gall, S. Katsikas, and K. Karapoulios. Application of Genetic Algorithms to Software Testing. In *Proceedings of the 5th International Conference on Software Engineering and Applications*, pages 625–636, Toulouse, France, 7-11 December 1992.

[ZFH08] Yuanyuan Zhang, Anthony Finkelstein, and Mark Harman. Search Based Requirements Optimisation: Existing Work & Challenges. In *Proceedings of the 14th International Working Conference, Requirements Engineering: Foundation for Software Quality (REFSQ '08)*, volume 5025 of *LNCS*, pages 88–94, Montpellier, France, 16-17 June 2008. Springer.