Claire Le Goues
Shin Yoo (Eds.)

LNCS 8636

Search-Based Software Engineering

6th International Symposium, SSBSE 2014
Fortaleza, Brazil, August 26–29, 2014
Proceedings

Springer

Lecture Notes in Computer Science 8636

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Claire Le Goues Shin Yoo (Eds.)

Search-Based Software Engineering

6th International Symposium, SSBSE 2014
Fortaleza, Brazil, August 26-29, 2014
Proceedings

Springer

Volume Editors

Claire Le Goues
Carnegie Mellon University
School of Computer Science
Institute for Software Research
5000 Forbes Avenue, Pittsburgh, PA 15213, USA
E-mail: clegoues@cs.cmu.edu

Shin Yoo
University College London
Department of Computer Science
Gower Street, London WC1E 6BT, UK
E-mail: shin.yoo@ucl.ac.uk

ISSN 0302-9743
e-ISSN 1611-3349
ISBN 978-3-319-09939-2
e-ISBN 978-3-319-09940-8
DOI 10.1007/978-3-319-09940-8
Springer Cham Heidelberg New York Dordrecht London

Library of Congress Control Number: 2014945224

LNCS Sublibrary: SL 2 – Programming and Software Engineering

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

Message from the SSBSE 2014 General Chair

SBSE is growing up! In its sixth edition, the conference left home and expanded its reach in a process of becoming a truly global forum. Brazil was proudly chosen to kick off this process, mainly in recognition of its strong and still growing SBSE community. Besides innovating in its location, SSBSE 2014 implemented a series of novelties. First, it stood alone once again. As a test of maturity, this decision sheds light on how independent and solid the SBSE field has become. Second, it brought an all-inclusive experience, allowing for a much higher level of integration among participants, in turn strengthening the community and helping create a much more cooperative environment. Finally, it implemented a double-blind submission and review process for the first time, providing as fair and objective an evaluation of the submitted papers as possible.

Obviously, this historical event would not have been possible without the help of many people, who I would like to recognize and thank. First of all, I would like to thank our program chairs, Shin Yoo (University College London, UK) and Claire Le Goues (Carnegie Mellon University, USA). They led the review process with great competence and dedication and put together a very rich and high-quality scientific program. I extend this recognition to all members of our Program Committee, for the dedicated work in the review and selection of our papers. Next, I thank our Graduate Student Track chair, Nadia Alshahwan (University College London, UK), and our SBSE Challenge Track chair (Márcio de Oliveira Barros, Federal University of the State of Rio de Janeiro, Brazil), for their hard work on organizing those two special tracks. I would also like to give special thanks to my friend Allysson Araújo (State University of Ceará, Brazil), our Web chair, for accepting the important challenge of creating and maintaining our website and operating this task with perfection. Also, I thank our publicity chair, Sina Shamshiri (University of Sheffield, UK), for the important job of keeping everybody informed about our event. Finally, I also thank the SSBSE Steering Committee, chaired by Mark Harman (University College London, UK), for their vote of confidence in giving us the privilege of organizing SSBSE 2014.

I must also mention and thank our long list of sponsors, who believed in our proposal and provided confidence in me and in the field of SBSE. Without their support, SSBSE 2014 would not have been nearly so special.

I hope you enjoy reading these proceedings as much as I enjoyed organizing the event.

August 2014 Jerffeson Teixeira de Souza

Message from the SSBSE 2014 Program Chairs

On behalf of the SSBSE 2014 Program Committee, we are pleased to present the proceedings of the 6th International Symposium on Search-Based Software Engineering. This year brought SSBSE to South America for the first time, in the oceanside paradise of Fortaleza, Brazil! SSBSE 2014 continued to bring together the international community of SSBSE researchers to exchange and discuss ideas and celebrate the latest progress in this rapidly advancing field.

We are delighted to report that we had a record-breaking 51 submissions to our four tracks: 32 Full Research Track submissions, eight Graduate Student Track submissions, three Fast Abstract submissions, and eight SBSE Challenge Track submissions. Submissions came from 19 different countries: Argentina, Austria, Brazil, Canada, China, the Czech Republic, France, Germany, India, Ireland, Italy, Luxembourg, Norway, the Russian Federation, Sweden, Switzerland, Tunisia, the UK, and the USA. After each submission was reviewed by at least three members of the Program Committee, we accepted 14 Full Research Track papers, one Fast Abstract track paper, three Graduate Student Track papers, and four SBSE Challenge Track papers.

We would like to thank the members of the SSBSE 2014 Program Committee. Without their continued support, we would not have been able to further improve the quality of the submissions and maintain the symposium's tradition of a high-quality technical program. The general chair, Jerffeson Teixeira de Souza, deserves a special mention for leading an excellent team, especially locally, to make the conference an unforgettable experience for everyone. In addition, Márcio Barros worked hard to manage the fast-growing SBSE Challenge Track, while Nadia Alshahwan oversaw the process of handling the Graduate Student Track. The technical program would not have been the same without their effort, for which we especially want to thank them.

As an experiment, this year we implemented a double-blind review procedure for the main research track of the SSBSE program. Our intention was to enable as fair a review process as possible, and recent evidence suggests that removing information like institutional affiliation, country of origin, and author name from submissions under review can contribute to this goal. We want to thank both the Program Committee and our submitting authors for their patience with a new and largely unfamiliar system, and for allowing us to experiment with our review procedure. We encourage those who participated to continue sharing their perspectives on this and other issues related to review and feedback quality. Peer review remains a collaborative and work-in-progress system, and we are interested in the community's experience to help inform future decisions for both this conference and others like it.

The symposium has an excellent tradition of hearing and learning from world experts in both software engineering and meta-heuristic optimization, and we are glad to report that this year was not an exception. We had the honor of having a keynote from Prof. Mauro Pezzè, whose research on software redundancy bears a strong connection to SBSE. Furthermore, we also had a keynote from Dr. Marc Schoenauer, who brought us up to date with progress in adaptive learning research. Finally, the Brazilian SBSE community warmly and enthusiastically invited Prof. Mark Harman to present a review of the field.

We would like to thank all the authors who submitted papers to SSBSE 2014, regardless of the outcome, and everyone who attended the symposium. We hope that, with these proceedings, anyone who did not have a chance to be at Fortaleza will have the opportunity to experience the exuberance of the SBSE community.

August 2014

Claire Le Goues
Shin Yoo

Conference Organization

General Chair

Jerffeson Teixeira de Souza	Universidade Estadual do Ceará, Brazil

Program Chairs

Claire Le Goues	Carnegie Mellon University, USA
Shin Yoo	University College London, UK

Graduate Students Track Chair

Nadia Alshahwan	University College London, UK

SBSE Challenge Track Chair

Márcio de Oliveira Barros	Universidade Federal do Estado do Rio de Janeiro, Brazil

Organizing Committee

Sina Shamshiri	University of Sheffield, UK
Allysson Allex de Paula Araújo	Universidade Estadual do Ceará, Brazil

Program Committee

Enrique Alba	University of Málaga, Spain
Shaukat Ali	Simula Research Laboratory, Norway
Giuliano Antoniol	Ecole Polytechnique de Montréal, Canada
Andrea Arcuri	Schlumberger & Simula Research Laboratory, Norway
Leonardo Bottaci	University of Hull, UK
Betty Cheng	Michigan State University, USA
Francisco Chicano	University of Málaga, Spain
Myra Cohen	University of Nebraska-Lincoln, USA
Massimiliano Di Penta	University of Sannio, Italy
Arilo Claudio Dias-Neto	Universidade Federal do Amazonas, Brazil

Robert Feldt	Blekinge Institute of Technology, Sweden
Gordon Fraser	University of Sheffield, UK
Mathew Hall	University of Sheffield, UK
Mark Harman	University College London, UK
Colin Johnson	University of Kent, UK
Gregory Kapfhammer	Allegheny College, USA
Marouane Kessentini	University of Michigan, USA
Dongsun Kim	University of Luxembourg, Luxembourg
Yvan Labiche	Carleton University, Canada
Raluca Lefticaru	University of Bucharest, Romania
Zheng Li	Beijing University of Chemical Technology, China
Spiros Mancoridis	Drexel University, USA
Auri Marcelo Rizzo Vincenzi	Universidade Federal de Goiás, Brazil
Tim Menzies	North Carolina State University, USA
Fitsum Meshesha Kifetew	Fondazione Bruno Kessler - IRST, Italy
Leandro Minku	University of Birmingham, UK
Martin Monperrus	University of Lille, France
Mel Ó Cinnéide	University College Dublin, Ireland
Justyna Petke	University College London, UK
Pasqualina Potena	University of Alcalá, Spain
Simon Poulding	University of York, UK
Xiao Qu	ABB Corporate Research, USA
Marc Roper	University of Strathclyde, UK
Federica Sarro	University College London, UK
Chris Simon	University of the West of England, UK
Lee Spector	Hampshire College, USA
Angelo Susi	Fondazione Bruno Kessler - IRST, Italy
Jerry Swan	University of Stirling, UK
Paolo Tonella	Fondazione Bruno Kessler - IRST, Italy
Silvia Vergilio	Universidade Federal do Paraná, Brazil
Tanja E.J. Vos	Universidad Politècnica de València, Spain
David White	University of Glasgow, UK
Xin Yao	University of Birmingham, UK

External Reviewers

Kenyo Faria	Universidade Federal de Goiás, Brazil
Eduardo Freitas	Universidade Federal de Goiás, Brazil
Damiano Torre	Carleton University, Canada
Shuai Wang	Simula Research Laboratory, Norway
Zhihong Xu	University of Nebraska Lincoln, USA

Steering Committee

Mark Harman (Chair)	University College London, UK
Andrea Arcuri	Schlumberger & Simula Research Laboratory, Norway
Massimiliano Di Penta	University of Sannio, Italy
Gordon Fraser	University of Sheffield, UK
Mel Ó Cinnéide	University College Dublin, Ireland
Jerffeson Teixeira de Souza	Universidade Estadual do Ceará, Brazil
Joachim Wegener	Berner and Mattner, Germany
David White	University of Glasgow, UK
Yuanyuan Zhang	University College London, UK

Sponsors

Keynote Addresses

Intrinsic Software Redundancy: Applications and Challenges (Extended Abstract)

Mauro Pezzè

University of Lugano, Switzerland and University of Milano Bicocca, Italy
mauro.pezze@usi.ch

Abstract. Search-based software engineering has many important applications. Here, we identify a novel use of search-based techniques to identify redundant components. Modern software systems are intrinsically redundant, and such redundancy finds many applications. In this paper we introduce the concept of intrinsic redundancy, and we present some important applications to develop self-healing systems and automatically generate semantically relevant oracles. We then illustrate how search-based software engineering can be used to automatically identify redundant methods in software systems, thus paving the road to an efficient exploitation of intrinsic redundancy, and opening new research frontiers for search-based software engineering.

Reliability is becoming a necessity for many software systems and redundancy is its cornerstone. Well defined processes, efficient design approaches, careful coding and pervasive testing and analysis can build excellent software products, but cannot completely eliminate failures in the field, and the software products may not meet a sufficient reliability level.

The classic way of improving the reliability of systems of different kinds exploits some form of redundancy. RAID disks (Redundant Array of Independent Disks) are a successful example of the use of redundancy for improving hardware reliability [1], the HDFS (Hadoop Distributed File System) is a popular example of the use of redundancy for improving data reliability [2], N-version programming is a classic approach that exploits redundancy for improving software reliability [3].

In these different approaches, redundancy is *deliberately* added to the system to improve reliability, and comes with additional costs that depend on the goals. In hardware systems, redundancy aims to reduce the impact of production defects, and is added at the production level, thus impacts mostly on production costs. In database systems, redundancy is added at the server level and impacts mostly on infrastructure costs. N-version programming targets design errors and is added at the design level, where the impact on costs is relevant.

We point to a different kind of software redundancy that is *intrinsically* present in software systems, and is thus available without additional design or production costs. Our empirical investigation indicates that such form of redundancy is widely spread in modern software systems and is a consequence of good design practice. Our work shows that this form of redundancy can be automatically synthetized by means of search-based techniques [4], and can be successfully exploited in many ways, including the automatic generation of self-healing mechanisms [5] and of semantic oracles [6].

Redundancy is present at many abstraction levels, here we discuss it referring to redundancy at method call level. We say that two methods are redundant when their execution is both different and produces equivalent results. Results are equivalent when both the output and the final state are indistinguishable from an external observer viewpoint, as formalised with the concept of observational equivalence [7]. Executions are different when they involve different statements or the same statements but in different order.

Redundancy is intrinsically present in software systems due to modern design practice. Design for reusability often leads to the same functionality implemented in different methods to improve compatibility with different uses, as it happened in containers that provide different methods to add one or more elements to the container. Performance optimisation frequently results in different methods implementing the same functionality, albeit with different, optimised code, like the *trove4J* library that duplicates many of the functionalities offered by the standard Java containers. Backward compatibility is obtained by keeping the old versions of the reimplemented functionalities thus offering redundant methods. Redundancy is massively present in modern software systems: Our manual inspection of several popular libraries including Apache Ant, Google Guava, Joda Time, Eclipse SWT, graphstream and Lucene identified over 4,700 redundant methods, with an average of 5 redundant methods per class.

Intrinsic redundancy can be exploited to build self-healing mechanisms. Once identified a set of redundant methods, we can automatically deploy a mechanism that substitutes a failing method with a redundant one to avoid the failure. We call such approach *automatic workaround*. The design of automatic workarounds requires a mechanism to reveal failures, we rely on assertions embedded in the code, a method to roll back to a correct state, we rely on an optimised rollback mechanism, and a method to execute a redundant method, we rely on a source to source code transformation [8, 5].

Another interesting application of intrinsic redundancy is the automatic synthesis of semantically relevant test oracles. The increasing availability of automated test cases exacerbates the need of automated oracles, and the cost pressure of software development calls for automatically generated oracles. Oracles that can be easily generated automatically, such as implicit oracles, can only reveal simple failures, like unhandled exceptions, while oracles derived from formal specifications can reveal failures that depend on the program semantics, but require formal specifications that are expensive to produce. We exploit the intrinsic redundancy of software systems to automatically generate test oracles that can reveal failures related to the program semantics by cloning the program state before executing a method call, executing the original call on the original state and the corresponding redundant call on the cloned state, and comparing the results. In this way we can reveal discrepancies between the executions of methods that should produce equivalent results and reveal failures related to the program semantics. We call such oracles *cross-checking oracles* [6].

The automatic synthesis of both self-healing mechanisms and automated oracles requires a set of redundant program elements as input. We can automatically synthetize redundant methods without expensive formal specifications by exploiting search-based techniques. We use a genetic algorithm for synthetizing a method call equivalent to a given method for an initial scenario (usually one or few test cases). We then look for a counterexample that, if found, gives us a new scenario to search for a redundant method, and, if not found, confirms the redundancy of the original and the identified method. We can automatically synthetize a large amount of redundant methods by applying the approach to all methods in the target software system.

Acknowledgement. We would like to acknowledge the Swiss National Foundation (SNF) for supporting this work through the projects Perseos (SNF 200021_116287), Wash (SNF 200020_124918) and Shade (SNF 200021_138006), and the many people who contributed to the work, Antonio Carzaniga, Alberto Goffi, Alessandra Gorla, Andrea Mattavelli, Nicolò Perino and Paolo Tonella.

References

1. Patterson, D., Gibson, G., Katz, R.: A case for redundant arrays of inexpensive disks (RAID). SIGMOD Record 17(3), 109–116 (1988)
2. Shvachko, K., Kuang, H., Radia, S., Chansler, R.: The hadoop distributed file system. In: Proceedings of the IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST), pp. 1–10. IEEE Computer Society (2010)
3. Avizienis, A.: The N-version approach to fault-tolerant software. IEEE Transactions on Software Engineering 11(12), 1491–1501 (1985)
4. Goffi, A., Gorla, A., Mattavelli, A., Pezzè, M., Tonella, P.: Search-based synthesis of equivalent method sequences. In: Proceedings of the 2014 ACM Sigsoft Conference on the Foundations of Software Engineering (ACM FSE). ACM (2014)
5. Carzaniga, A., Gorla, A., Mattavelli, A., Perino, N., Pezzè, M.: Automatic recovery from runtime failures. In: Proceedings of the 2013 International Conference on Software Engineering (ICSE), pp. 782–791. IEEE Press (2013)
6. Carzaniga, A., Goffi, A., Gorla, A., Mattavelli, A., Pezzè, M.: Cross-checking oracles from intrinsic software redundancy. In: Proceedings of the 2014 International Conference on Software Engineering (ICSE). IEEE Press (2014)
7. Hennessy, M., Milner, R.: On observing nondeterminism and concurrency. In: de Bakker, J., van Leeuwen, J. (eds.) ICALP 1980. LNCS, vol. 85, pp. 299–309. Springer, Heidelberg (1980)
8. Carzaniga, A., Gorla, A., Perino, N., Pezzè, M.: Automatic workarounds for web applications. In: Proceedings of the 2010 ACM Sigsoft Conference on the Foundations of Software Engineering (ACM FSE), pp. 237–246. ACM (2010)

Programming by Ranking (Extended Abstract)

Marc Schoenauer[1,2] and Michèle Sebag[2,1]

[1] TAO Project-team, INRIA Saclay - Île-de-France
[2] Laboratoire de Recherche en Informatique (UMR CNRS 8623)
Université Paris-Sud, 91128 Orsay Cedex, France
FirstName.LastName@inria.fr

As early as 1904, Spearman [19] proposed to use ranks rather than actual values to unveil correlations between data of unknown distribution. This was the beginning of rank statistics and non-parametric tests. Still, in practice non-parametric statistics are generally less accurate than their parametric counterparts (even though more widely applicable), and the latter are often used even though the underlying hypotheses (normally distributed data, size of sample) are not satisfied.

In the context of optimization and programming however, rank-based approaches might prove more beneficial that value-based approaches even in cases where both approaches apply. Three test cases related to Algorithm Engineering will be surveyed here, dealing with Black-Box Optimization (Section 1), Algorithm Selection using ideas from recommender systems (Section 2) and robot programming by weak experts (Section 3).

1 Rank-SVM Surrogate Models for CMA-ES

In the general framework of (black-box) continuous optimization, the human mind cannot always easily grasp quantified measures to assess the quality of a potential solution. In a famous example of interactive optimization [8], when the coffee maker asks some coffee experts how close the taste of a coffee is from a targeted coffee taste, there does not even exist a scale that could be used by all experts to put a number on taste proximity. However, every expert can assess whether a coffee is closer or farther than another one with respect to the target taste.

Quantifying differences might be a problem even when there is no human being in the loop. More generally, the optimization of a given real-valued function $\mathcal{F}$ is unchanged if $\mathcal{F}$ undergoes any monotonous transformation (from preconditioning to non-linear regularization), although this can have a huge impact on the efficiency of most optimization algorithms. Comparison-based algorithms de facto possess such invariance w.r.t. monotonous transformations. In the case of expensive optimization problems, the usual strategy is to learn an approximation a.k.a. surrogate model of $\mathcal{F}$ by numerical regression; this strategy however destroys the invariance property as the surrogate model depends on the *values* of $\mathcal{F}$. Ordinal regression, aka rank-based learning, instead defines a surrogate model which only preserves the ranks of the $\mathcal{F}$ values [10]. The use of such rank-based surrogates preserves the invariance property in comparison-based optimization [17]. Interestingly, the Covariance Matrix Adaptation Evolution Strategy [7] (CMA-ES, considered today the state-of-the-art method in Black-Box optimization) can be tightly integrated with rank-based learning, thus preserving all invariance properties of CMA-ES [14], while enforcing the control and adaptation of the learning hyper-parameters [15]. The resulting surrogate-augmented algorithm further improves the performance of the basic variants of CMA-ES on the BBOB (Black-Box Optimization Benchmarking [5]) platform.

2 Algorithm Selection as a Collaborative Filtering

In the domain of recommendation algorithms, similarly, movie rating can vary a lot from user to user, clearly raising a scaling issue in recommendation systems [4]. On the opposite, any user is able to rank the movies he has seen. The CofiRank method [21] uses the Maximal Margin Matrix Factorization to approximate rankings rather than ratings, bringing more robustness in the recommender system.

Similar issues arise in algorithm selection, a key issue to get peak performance from algorithm portfolios. It turns out that algorithm selection can be formalized as a collaborative filtering problem [20], by considering that a problem instance "prefers" the algorithms with better performance on this particular instance. Learning the *rating*, i.e. the actual performance of the algorithm on the problem instance raises significant difficulties, as the performance of an algorithm can vary by orders of magnitude depending on the problem instance. Learning how to rank algorithms depending on the problem instance can instead be achieved efficiently [16].

A main difficulty in algorithm selection is the handling of the so-called 'cold start' problem: how to choose an algorithm for a brand-new instance? Former algorithm selection methods relied on known features describing the problem instances – however with mixed results [12]. But the Matrix Factorization method amounts to identify latent features that are by construction well suited to the algorithm selection problem. Supervised learning of a mapping between known features and those latent features is the key to solving the cold-start problem, as demonstrated in [16] on three problem domains (the 2008 CSP and 2011 SAT competitions, and the BBOB platform).

3 Programming by Feedback

In the context of adapting software or hardware agents (e.g., a companion robot) to the precise requirements or preferences of their human users, the limitation comes from both the quantity and the quality of what can be asked to the user. Whereas you can ask experts to demonstrate the desired behavior to the robot, as in Inverse Reinforcement Learning approaches [1, 11], you can only ask limited amount of feedback to the average user. On the one hand, the feedback is uneasily provided through numbers; on the other hand, even a preference feedback (it is better, it is worse) can be noisy and inconsistent.

Preference-based reinforcement learning, hybridizing reinforcement learning and learning to rank, has been proposed to handle domains with no numerical rewards [6], allowing the user to compare and rank the agent behaviors [22, 2]. The key issues are to deliver a good performance with a limited number of comparison queries, and particularly to stand the comparison errors and the possible user inconsistencies. These issues have been addressed in [3], enabling the agent to model the user's competence; indeed the cooperation between two intelligent partners is better supported by each partner having a model of the other one (see e.g. [13]).

From the user point of view, the game is similar to the well-known children's game "Hot-and-Cold": she only has to tell the robot whether each new demonstrated behavior is better or worse than the previous one – and she can be inconsistent (or her goal can evolve). From the robot perspective, the idea is to gradually learn the user's utility function in the demonstration space, accounting for the user's estimated competence, and, based on the current utility function, to optimize in the solution space the behavior with respect to some maximal posterior utility, demonstrating the best one to the user. Experimental results on artificial RL benchmark problems favorably compare to the state of the art [22], and proof-of-principle results are obtained on a real NAO robot, though on elementary tasks: 5 (resp. 24) interactions with the user are required to solve problems involving 13 (resp. 20) states spaces.

4 Conclusion

There is emerging evidence that the art of programming could be revisited in the light of the current state of the art in Machine Learning and Optimization. While the notion of formal specifications has been at the core of software sciences for over three decades, the relevance of ML-based approaches has been demonstrated in the domain of pattern recognition since the early 90s.

Along this line, a new trend dubbed *Programming by Optimization* advocates algorithm portfolios endowed with a control layer such that *determining what works best in a given use context [could be] performed automatically, substituting human labor by computation* [9]. Similarly, it has been suggested that *the state of the art can be improved by configuring existing techniques better rather than inventing new learning paradigms* [18].

Going further, we propose the *Programming by Ranking* paradigm, extending the above *Programming by Feedback*; several proofs of principle thereof in different domains have been described, related to expensive optimization, algorithm selection and policy design. Ultimately, our claim is that learning-to-rank Machine Learning algorithms, based on minimal and possibly noisy specification/information/feedback from the user, have today reached the come-of-age and should be considered whenever optimization at large is at stake.

References

1. Abbeel, P.: Apprenticeship Learning and Reinforcement Learning with Application to Robotic Control. PhD thesis, Stanford University (2008)
2. Akrour, R., Schoenauer, M., Sebag, M.: April: Active preference learning-based reinforcement learning. In: Flach, P.A., De Bie, T., Cristianini, N. (eds.) ECML PKDD 2012, Part II. LNCS, vol. 7524, pp. 116–131. Springer, Heidelberg (2012)
3. Akrour, R., Schoenauer, M., Sebag, M., Souplet, J.-C.: Programming by feedback. In: ICML. volume to appear of ACM Int. Conf. Proc. Series (2014)
4. Bennett, J., Lanning, S.: The netflix prize. In: Proc. 13th Intl Conf. on Knowledge Discovery and Data Mining (KDD) Cup and Workshop (2007)
5. Finck, S., Hansen, N., Ros, R., Auger, A.: Real-parameter black-box optimization benchmarking 2010: Experimental setup. Technical Report 2009/21, Research Center PPE (2010)
6. Fürnkranz, J., Hüllermeier, E., Cheng, W., Park, S.-H.: Preference-based reinforcement learning. Machine Learning 89(1-2), 123–156 (2012)
7. Hansen, N., Müller, S., Koumoutsakos, P.: Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). Evolutionary Computation 11(1), 1–18 (2003)
8. Herdy, M.: Evolution strategies with subjective selection. In: Ebeling, W., Rechenberg, I., Voigt, H.-M., Schwefel, H.-P. (eds.) PPSN IV. LNCS, vol. 1141, pp. 22–31. Springer, Heidelberg (1996)
9. Hoos, H.H.: Programming by optimization. Commun. ACM 55(2), 70–80 (2012)
10. Joachims, T.: A support vector method for multivariate performance measures. In: Proceedings of the 22nd International Conference on Machine Learning, pp. 377–384. ACM (2005)
11. Konidaris, G., Kuindersma, S., Barto, A., Grupen, R.: Constructing skill trees for reinforcement learning agents from demonstration trajectories. In: NIPS 23, pp. 1162–1170 (2010)
12. Kotthoff, L.: Hybrid regression-classification models for algorithm selection. In: De Raedt, L., et al. (eds.) Proc. ECAI 2012, pp. 480–485. IOS Press (2012)
13. Lörincz, A., Gyenes, V., Kiszlinger, M., Szita, I.: Mind model seems necessary for the emergence of communication. Neural Information Processing - Letters and Reviews 11(4-6), 109–121 (2007)
14. Loshchilov, I., Schoenauer, M., Sebag, M.: Comparison-Based Optimizers Need Comparison-Based Surrogates. In: Schaefer, R., Cotta, C., Kołodziej, J., Rudolph, G. (eds.) PPSN XI. LNCS, vol. 6238, pp. 364–373. Springer, Heidelberg (2010)
15. Loshchilov, I., Schoenauer, M., Sebag, M.: Self-Adaptive Surrogate-Assisted cma-es. In: ACM-GECCO, pp. 321–328. ACM Press (July 2012)
16. Misir, M., Sebag, M.: Algorithm selection as a collaborative filtering problem. Technical report, INRIA (December 2013)
17. Runarsson, T.P.: Ordinal Regression in Evolutionary Computation. In: Runarsson, T.P., Beyer, H.-G., Burke, E., Merelo-Guervós, J.J., Whitley, L.D., Yao, X. (eds.) PPSN IX. LNCS, vol. 4193, pp. 1048–1057. Springer, Heidelberg (2006)
18. Snoek, J., Larochelle, H., Adams, R.P.: Practical Bayesian optimization of machine learning algorithms. In: NIPS, pp. 2960–2968 (2012)

19. Spearman, C.: The proof and measurement of association between two things. The American Journal of Psychology 100(3-4), 441–471 (1904)
20. Stern, D., Herbrich, R., Graepel, T., Samulowitz, H., Pulina, L., Tacchella, A.: Collaborative expert portfolio management. In: AAAI, pp. 179–184 (2010)
21. Weimer, M., Karatzoglou, A., Le, Q., Smola, A.: COFI-RANK: Maximum Margin Matrix Factorization for Collaborative Ranking. In: NIPS 2007, pp. 222–230 (2007)
22. Wilson, A., Fern, A., Tadepalli, P.: A Bayesian approach for policy learning from trajectory preference queries. In: NIPS, pp. 1142–1150 (2012)