Instance-Specific Algorithm Configuration

Yuri Malitsky

Instance-Specific Algorithm Configuration

Springer

Yuri Malitsky
IBM Thomas J. Watson Research Center
Yorktown Heights
New York
USA

ISBN 978-3-319-11229-9 ISBN 978-3-319-11230-5 (eBook)
DOI 10.1007/978-3-319-11230-5
Springer Cham Heidelberg New York Dordrecht London

Library of Congress Control Number: 2014956556

Printed on acid-free paper

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

Preface

When developing a new heuristic or complete algorithm for a constraint satisfaction or constrained optimization problem, we frequently face the problem of choice. There may be multiple branching heuristics that we can employ, different types of inference mechanisms, various restart strategies, or a multitude of neighborhoods from which to choose. Furthermore, the way in which the choices we make affect one another is not readily perceptible. The task of making these choices is known as algorithm configuration.

Developers often make many of these algorithmic choices during the prototyping stage. Based on a few preliminary manual tests, certain algorithmic components are discarded even before all the remaining components have been implemented. However, by making the algorithmic choices beforehand developers may unknowingly discard components that are used in the optimal configuration. In addition, the developer of an algorithm has limited knowledge about the instances that a user will typically employ the solver for. That is the very reason why solvers have parameters: to enable users to fine-tune a solver for their specific needs.

Alternatively, manually tuning a parameterized solver can require significant resources, effort, and expert knowledge. Before even trying the numerous possible parameter settings, the user must learn about the inner workings of the solver to understand what each parameter does. Furthermore, it has been shown that manual tuning often leads to highly inferior performance.

This book shows how to automatically train a multi-scale, multi-task approach for enhanced performance based on machine learning techniques. In particular this work presents the methodology of Instance-Specific Algorithm Configuration (ISAC). ISAC is a general configurator that focuses on tuning different categories of parameterized solvers according to the instances they will be applied to. Specifically, this book shows that the instances of many problems can be decomposed into a representative vector of features. It further shows that instances with similar features often cause similar behavior in the applied algorithm. ISAC exploits this observation by automatically detecting the different subtypes of a problem and then training a solver for each variety. This technique is explored on a number of problem domains, including set covering, mixed integer, satisfiability, and set partitioning. ISAC is then further expanded to demonstrate its application to traditional algorithm portfolios and adaptive search methodologies. In all cases, marked improvements are shown over the existing state-of-the-art solvers. These improvements were particularly evident during the 2011 SAT Competition, where a solver based on ISAC won seven medals, including a gold in the handcrafted instance category, and another gold in the randomly generated instance category. Later, in 2013, solvers based on this research won multiple gold medals in both the SAT Competition and the MaxSAT Evaluation.

The research behind ISAC is broken down into ten chapters. Chapters 1 and 2, respectively, introduce the problem domain and the relevant research that has been done on the topic. Chapter 3 then introduces the ISAC methodology, while Chap. 4 demonstrates its effectiveness in practice. Chapters 5, 6, and 7 demonstrate how the methodology can be applied to the problem of algorithm selection. Chapter 8 takes an alternate view and shows how ISAC can be used to create a framework that dynamically switches to the best heuristic to utilize as the problem is being solved. Chapter 9 introduces the concept that sometimes problems change over time, and that a portfolio needs a way to effectively retrain to accommodate the changes. Chapter 10 demonstrates how the ISAC methodology can be transparently expanded, while Chap. 11 wraps up the topics covered and offers ideas for future research.

The research presented in this book was carried out as the author's Ph.D. work at Brown University and his continuing work at the Cork Constraint Computation Centre. It would not have been possible without the collaboration with my advisor, Meinolf Sellmann, and my supervisor, Barry O'Sullivan. I would also like to thank all of my coauthors, who have helped to make this research possible. In alphabetical order they are: Tinus Abell, Carlos Ansotegui, Marco Collautti, Barry Hurley, Serdar Kadioglu, Lars Kotthoff, Christian Kroer, Kevin Leo, Giovanni Di Liberto, Deepak Mehta, Ashish Sabharwal, Horst Samulowitz, Helmut Simonis, and Kevin Tierney.

This work has also been partially supported by Science Foundation Ireland Grant No. 10/IN.1/I3032 and by the European Union FET grant (ICON) No. 284715.

Cork, Ireland Yuri Malitsky
September 2013

Contents