

SpringerBriefs in Computer Science

Series editors

- Stan Zdonik, Brown University, Providence, USA
Shashi Shekhar, University of Minnesota, Minneapolis, USA
Jonathan Katz, University of Maryland, College Park, USA
Xindong Wu, University of Vermont, Burlington, USA
Lakhmi C. Jain, University of South Australia, Adelaide, Australia
David Padua, University of Illinois Urbana-Champaign, Urbana, USA
Xuemin (Sherman) Shen, University of Waterloo, Waterloo, Canada
Borko Furht, Florida Atlantic University, Boca Raton, USA
V.S. Subrahmanian, University of Maryland, College Park, USA
Martial Hebert, Carnegie Mellon University, Pittsburgh, USA
Katsushi Ikeuchi, University of Tokyo, Tokyo, Japan
Bruno Siciliano, Università di Napoli Federico II, Napoli, Italy
Sushil Jajodia, George Mason University, Fairfax, USA
Newton Lee, Newton Lee Laboratories, LLC, Tujunga, USA

More information about this series at <http://www.springer.com/series/10028>

Jian Zhang · Zhiqiang Zhang
Feifei Ma

Automatic Generation of Combinatorial Test Data

Jian Zhang
Zhiqiang Zhang
Feifei Ma
Institute of Software
Chinese Academy of Sciences
Beijing
China

ISSN 2191-5768 ISSN 2191-5776 (electronic)
ISBN 978-3-662-43428-4 ISBN 978-3-662-43429-1 (eBook)
DOI 10.1007/978-3-662-43429-1

Library of Congress Control Number: 2014948067

Springer Heidelberg New York Dordrecht London

© The Author(s) 2014

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed. Exempted from this legal reservation are brief excerpts in connection with reviews or scholarly analysis or material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work. Duplication of this publication or parts thereof is permitted only under the provisions of the Copyright Law of the Publisher's location, in its current version, and permission for use must always be obtained from Springer. Permissions for use may be obtained through RightsLink at the Copyright Clearance Center. Violations are liable to prosecution under the respective Copyright Law. The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Printed on acid-free paper

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

Foreword

When software is designed using a large number of components, and is then executed in an environment that itself contains a large number of components, faults often arise from unexpected interactions among the components. How can we test these complex systems of interacting components? In particular, how can we find test suites that can reveal the most likely faults? The combinatorial explosion in the number of possible tests must be avoided; in principle, it can be avoided if faults arise from interactions among a few components. Yet generating small test suites remains a very challenging problem.

Some research concentrates on the underlying combinatorial problems and neglects the needs of the software tester; other research addresses testing without exploiting combinatorial information effectively. This book is valuable because it encompasses both the combinatorial theory and the testing practice. It formulates the construction of test suites as a combinatorial problem, clearly explaining the connection with testing needs. After introducing important relationships with studies in combinatorics, it focusses on algorithmic methods for the construction of test suites. The blending of software design, algorithms, and combinatorial mathematics provides an excellent introduction to this important topic.

Phoenix, AZ, USA, June 2014

Charles J. Colbourn

Preface

Testing is an important part of the software development life cycle. Many testing techniques have been proposed and used, to increase the quality and reliability of software and systems. Among them, combinatorial testing is a kind of black-box testing method that is gaining increasing popularity. It is particularly useful to reveal faults that are caused by the interaction of a few components/parameters of the system.

For many kinds of testing techniques, a challenging problem is to generate a small set of test cases which achieves certain coverage criterion. This is also true for combinatorial testing. Over the past 30 years or so, many different kinds of algorithms have been proposed. Some of them were designed specifically to solve the problem, while others are based on ideas from evolutionary computation and constraint processing communities.

This book intends to review the state of the art of automatic data generation methods for combinatorial testing. The first chapter introduces the basic concepts and notations that will be used later. It also gives a brief overview of the applications of combinatorial testing. The next five chapters describe various kinds of methods for generating input data for combinatorial testing. Each of these chapters can be read almost independently. Chapter 7 gives a list of tools that can be readily used. In the last chapter, we briefly discuss several other problems that are closely related to test data generation.

The book can be read by graduate students and researchers who work in the area of combinatorial testing (or testing in general). It should also be interesting to those who work in discrete mathematics, constraint solving, evolutionary computation, and other related fields. Practitioners may find the book useful when selecting a suitable technique and tool.

Due to the limit on the size of the book, we could not explain every algorithm in detail. We may also miss some important works in this field. You are welcome to contact us if you have any comments and suggestions.

Our research on software testing is partially supported by the 973 Program of China (grant No. 2014CB340701) and the National Science Foundation of China (NSFC grant No. 61100064). We are indebted to Charles Colbourn, Jacek

Czerkwonka, Lijun Ji, Shanghai Nie, Zhilei Ren, Ziyuan Wang, Huayao Wu, Hantao Zhang, and Lie Zhu who read earlier drafts of this book and made many good suggestions. We would like to thank Jun Yan for his assistance during the preparation of this book. Thanks to Jeff Lei and Linbin Yu for answering our questions on the IPO algorithms. The editors at Springer, Celine Chang and Jane Li, are always helpful. We are grateful to them for their constructive suggestions and patience. In addition, the first author would like to thank Alfred Hofmann for his encouragement. Finally, we are very grateful to our families for their support.

Beijing, June 2014

Jian Zhang
Zhiqiang Zhang
Feifei Ma

Contents

1	Introduction to Combinatorial Testing	1
1.1	Software Testing	1
1.2	Combinatorial Testing and Its Applications	1
1.3	Related Combinatorial Designs	4
1.3.1	Latin Squares	4
1.3.2	Orthogonal Arrays	5
1.3.3	Covering Arrays	6
1.4	Covering Arrays for Testing	8
1.4.1	Seeding	8
1.4.2	Variable Strength Covering Arrays and Tuple Density	9
1.4.3	Covering Arrays with Constraints	10
1.4.4	Covering Arrays with Shielding Parameters	11
1.4.5	Sequence Covering Arrays	11
1.4.6	An Example of Using Covering Arrays for Testing	11
1.5	Pointers to the Literature	13
1.6	Structure of This Book	13
	References	14
2	Mathematical Construction Methods	17
2.1	Mathematical Methods for Constructing Orthogonal Arrays	17
2.1.1	Juxtaposition	17
2.1.2	Splitting	18
2.1.3	Hadamard Construction	18
2.1.4	Zero-Sum Construction	19
2.1.5	Construction from Mutually Orthogonal Latin Squares	19
2.2	Mathematical Methods for Constructing Covering Arrays	20
2.2.1	Simple Constructions	20
2.2.2	Recursive Constructions	21
2.2.3	Construction Based on Difference Covering Arrays	23
	References	25

3 One Test at a Time	27
3.1 General Idea of the One-Test Strategy	27
3.2 Automatic Efficient Test Generator (AETG)	28
3.2.1 The AETG Algorithm	28
3.2.2 AETG Example	29
3.3 AETG Variants	34
3.4 Test Generation Using Constrained Optimization	35
3.4.1 Example of PBO-Based Algorithm	37
References	39
4 The IPO Family	41
4.1 In-Parameter-Order.	41
4.2 IPOG-C for Handling Constraints	43
4.3 IPOG-C Example.	43
4.4 IPO Variants	48
References	49
5 Evolutionary Computation and Metaheuristics	51
5.1 Methods for Solving Optimization Problems	51
5.2 Applying Evolutionary Algorithms to Test Generation	52
5.3 Genetic Algorithms	53
5.3.1 Applying GA to Obtain the Whole Covering Array	53
5.3.2 Applying GA to Obtain a Good Test Case	54
5.4 Simulated Annealing	55
5.5 Particle Swarm Optimization	56
5.6 Tabu Search	57
5.7 Other Methods	58
References	58
6 Backtracking Search	61
6.1 Constraint Satisfaction Problems	61
6.1.1 Backtracking Algorithms for Solving CSPs.	62
6.1.2 Constraint Propagation.	62
6.1.3 Symmetry Breaking.	63
6.1.4 Constraint Solving Tools	64
6.2 Constraint Solving and the Generation of Covering Arrays	64
6.3 Backtracking Search for Generating Covering Arrays	65
6.3.1 Preprocessing	65
6.3.2 The Search Procedure	65
6.3.3 Exploiting Symmetries in CA	66
6.3.4 Sub-combination Equalization Heuristic	68
6.3.5 Combination of Strategies	69
6.3.6 Constraint Propagation.	70
6.3.7 An Example of the Search Process	71
6.3.8 A New Smaller CA	72

6.4	Backtracking Search for Generating Orthogonal Arrays	73
6.4.1	Preprocessing	73
6.4.2	The Basic Procedure	74
6.4.3	Exploiting Symmetries	75
	References	77
7	Tools and Benchmarks	79
7.1	Test Input Generation Tools	79
7.2	Applications and Benchmarks	80
	References	82
8	Other Related Topics	85
8.1	Problems Related to Test Input Generation	85
8.2	Modeling	86
8.3	Test Case Prioritization and Test Suite Reduction	87
8.4	Fault Localization	87
	References	89