

Contents

- Abstract I
- Acknowledgments VII
- List of Figures XIII
- List of Tables XVII
- List of Algorithms XXII
- List of Symbols XXIII
- List of Acronyms XXV
- List of Publications XXIX

- 1 Introduction 1
 - 1.1 Software testing overview 1
 - 1.2 Combinatorial interaction testing (CIT) 4
 - 1.3 Research problem 8
 - 1.3.1 Hypothesis 10
 - 1.3.2 Objective 10
 - 1.4 Contributions 10
 - 1.5 Thesis organization 10

2	Theoretical Framework	13
2.1	Latin Square and Orthogonal Latin Squares	13
2.2	Orthogonal Arrays	16
2.3	Covering Arrays	18
2.4	Summary	23
3	State of the Art	25
3.1	Computational complexity	25
3.2	Algebraic methods	26
3.2.1	Constructing optimal covering arrays within polynomial time	26
3.2.2	Group construction of covering arrays	27
3.2.3	Constant weight vectors	29
3.2.4	Using trinomial coefficients	31
3.3	Recursive methods	33
3.3.1	Raise to a power the number of columns for any alphabet and any strength	33
3.3.2	Products of covering arrays	35
3.3.3	Roux type constructions	35
3.4	Greedy methods	41
3.4.1	Automatic Efficient Test Generator (AETG)	41
3.4.2	Test Case Generation (TCG)	41
3.4.3	Deterministic Density Algorithm (DDA)	42
3.4.4	In-parameter-order (IPO)	44
3.4.5	Building-Block Algorithm (BBA)	46
3.4.6	Intersection Residual Pair Set Strategy (IRPS)	47
3.5	Metaheuristic methods	49
3.5.1	Tabu search (TS)	50
3.5.2	Ant colony optimization (ACO)	52
3.5.3	Simulated annealing (SA)	52
3.6	Construction of orthogonal arrays of index unity using logarithm tables for Galois fields	54
3.6.1	The Bush's construction	55
3.6.2	Algorithm for the construction of logarithm tables of Galois fields	56
3.6.3	Efficient construction of orthogonal arrays	57
3.6.4	Efficient constructions of covering arrays	60

3.7 Verification of covering arrays	62
3.7.1 Sequential algorithm to verify covering arrays	65
3.7.2 Parallel approach to verify covering arrays	66
3.7.3 Grid approach to verify covering arrays	71
3.8 Summary	75
4 Methodology	77
4.1 Simulated annealing overview	77
4.2 An improved simulated annealing to construct covering arrays	79
4.2.1 Internal representation.	79
4.2.2 Initial solution.	80
4.2.3 Evaluation function	82
4.2.4 Neighborhood function	84
4.2.5 Cooling schedule	84
4.2.6 Termination condition	85
4.2.7 Simulated annealing pseudocode.	85
4.3 Grid approach	86
4.3.1 Grid computing platform	86
4.3.2 Preprocessing task: selecting the most appropriate compute elements	87
4.3.3 Asynchronous schema	87
4.3.4 Synchronous schema	89
4.4 Parallel simulated annealing	90
4.4.1 Independent search approach.	91
4.4.2 Semi-independent search approach	91
4.4.3 Cooperative search approach	92
4.5 Summary	92
5 Experimental results	93
5.1 Analyzing the performance of simulated annealing	93
5.1.1 Influence of the initial solution.	94
5.1.2 Influence of the neighborhood functions	94
5.2 Fine tuning of the neighborhood functions	96
5.3 Sequential simulated annealing	98
5.3.1 Uniform covering arrays	98
5.3.2 Mixed covering arrays	99

5.4	Grid simulated annealing	101
5.5	Parallel simulated annealing	105
5.5.1	Comparison of the ISA, SSA and CSA approaches	105
5.5.2	Comparing the CSA approach against the state-of-the-art procedures	106
5.6	Constructing test-suites for different real-case software components	110
5.6.1	Case 1: Test Suites for a Smartphone Application	111
5.6.2	Case 2: Test Suites for the module <code>Add park</code>	114
5.7	Summary	115
6	Conclusions	119
6.1	Summary	119
6.2	Future work	121
6.3	Publications	122
	References	123
A	Covering arrays repository	137
A.1	Repositories for covering arrays	138
A.2	CINVESTAV covering arrays repository	140
A.2.1	Algorithms	140
A.2.2	Repository description	140
A.2.3	Scope and upper bounds of the repository	141
A.3	Cases of study	143
A.3.1	Algebraic constructions	144
A.3.2	Metaheuristics	144
A.4	Conclusions	145
	Index	147