

Contents

Part I Theory and Algorithms

1	Parallel Satisfiability	3
	Tomáš Balyo and Carsten Sinz	
1.1	Introduction	3
1.2	Preliminaries	4
1.2.1	Satisfiability (SAT)	4
1.2.2	Local Search Algorithms for SAT	5
1.2.3	The DPLL Algorithm	5
1.2.4	Resolution Refutation	7
1.2.5	The CDCL Algorithm	7
1.2.6	Parallel Computing Architectures	9
1.2.7	Measuring Speedups	9
1.3	Divide-and-Conquer Approaches	10
1.3.1	Problem Decomposition and Load Balancing	10
1.3.2	Implementations of Search-Space-Splitting Solvers	13
1.3.3	Search Space Splitting in CDCL	14
1.3.4	Cube and Conquer	15
1.4	Parallel Portfolios – Diversify and Conquer!	15
1.4.1	Virtual Best Solver	15
1.4.2	Pure Portfolio Solvers and Diversification	16
1.4.3	Clause-Sharing Portfolios	16
1.4.4	Impact of Diversification and Clause Sharing	17
1.4.5	Examples of Parallel Portfolio Solvers	19
1.5	Parallel Local Search	23
1.5.1	Multiple Flips	23
1.5.2	Portfolios	24
1.6	Future Challenges	24
	References	25

2	Cube-and-Conquer for Satisfiability	31	3.3.3.2	Clause-Sharing Heuristics	82
	Marijn J.H. Heule, Oliver Kullmann, and Armin Biere		3.3.3.3	Comparison Between Clause-Sharing Heuristics	84
2.1	Introduction	31	3.4	Deterministic Parallel MaxSAT	85
2.2	Preliminaries	33	3.4.1	Motivation	86
2.3	Combining CDCL and Lookahead	33	3.4.2	Deterministic Solver	87
2.4	Creating Cubes: The Basic Method	36	3.4.2.1	Standard Synchronization	89
2.5	Creating Cubes: a General Methodology	38	3.4.2.2	Period Synchronization	90
2.5.1	General Framework	38	3.4.2.3	Dynamic Synchronization	91
2.5.2	Cutoff Heuristic	39	3.4.3	Comparison Between Non-deterministic and Deterministic Solvers	92
2.5.3	Heuristics for Splitting	41	3.5	Research Directions	92
2.6	Solving Cubes	42	3.5.1	Scalability	92
2.6.1	Sequential Solving	42	3.5.2	Clause Sharing	93
2.6.2	Solving Cubes in Parallel	44		References	94
2.7	Interleaving the Cube and Conquer Phases	45	4	Parallel Solving of Quantified Boolean Formulas	101
2.7.1	Ineffective Lookahead Heuristics	46		Florian Lonsing and Martina Seidl	
2.7.2	Concurrent Cube-and-Conquer	46	4.1	Introduction	101
2.7.3	Cubes on Demand	48	4.2	Background	105
2.8	Proofs of Unsatisfiability	50	4.3	Sequential Search-Based QBF Solving	108
2.9	Experimental Evaluation	52	4.4	Parallel QBF Solving at a Glance	111
2.9.1	Application Benchmarks	52	4.5	Parallel QBF-Solving Approaches	115
2.9.2	The Boolean Pythagorean Triples Problem	54	4.6	Challenges and Potential of Parallel QBF Solving	127
2.10	Conclusions	56	4.7	Conclusion	131
	References	57		References	132
3	Parallel Maximum Satisfiability	61	5	Parallel Satisfiability Modulo Theories	141
	Inês Lynce, Vasco Manquinho, and Ruben Martins			Antti E.J. Hyvärinen and Christoph M. Wintersteiger	
3.1	Introduction	61	5.1	Introduction	141
3.2	Maximum Satisfiability	65	5.2	General Preliminaries	142
3.2.1	Sequential MaxSAT Algorithms	66	5.2.1	Theories	142
3.2.1.1	Linear Search Algorithms	67	5.2.2	The Underlying Conflict-Driven, Clause-Learning SAT Solver	143
3.2.1.2	Unsatisfiability-Based Algorithms	69	5.2.3	Theory Combination	144
3.2.1.3	Other Algorithmic Solutions and Implementation Issues	71	5.2.4	Interpolants	145
3.3	Parallel MaxSAT	71	5.2.5	SMT Solvers	145
3.3.1	Portfolio Approaches	72	5.3	Portfolios of SMT Solvers	146
3.3.1.1	Parallel Unsatisfiability-Based Algorithms	73	5.3.1	Parallel SMT Based on Algorithm Portfolios	148
3.3.1.2	Parallel Linear Search Algorithms	74	5.3.2	Lemma Sharing in Portfolios	148
3.3.1.3	Implementation Issues	75	5.3.3	Centralized Lemma Databases	149
3.3.2	Search Space Splitting	75	5.3.4	Experiments on the Algorithmic Framework	150
3.3.2.1	Interval Splitting	75	5.3.5	Lemma Sharing and Partitioning	150
3.3.2.2	Guiding Paths	79	5.4	Search-Space Partitioning in SMT	151
3.3.2.3	Other Splitting Schemes and Implementation Issues	80	5.4.1	Plain Partitioning	152
3.3.3	Clause Sharing	81	5.5	Decomposition	157
3.3.3.1	Conditions for Safe Clause Sharing	81	5.5.1	Experimental Evidence	158

5.5.2	Variations and Extensions	159
5.6	Combinations of Parallelization Algorithms	161
5.6.1	The Parallelization Tree	161
5.6.2	Iterative Partitioning with Partition Trees	163
5.6.3	Safe and Repeated Partitioning	165
5.6.4	Constructing Partitions	168
5.7	Further topics	171
	References	172
6	Parallel Theorem Proving	179
6.1	Maria Paola Bonacina	
6.1	Introduction	179
6.2	Theorem Proving Strategies	181
6.2.1	Subgoal-Reduction Strategies	182
6.2.2	Ordering-Based Strategies	184
6.2.2.1	Expansion-Oriented Strategies	186
6.2.2.2	Contraction-Based Strategies	187
6.2.3	Instance-Based Strategies	189
6.3	Parallelization of Theorem Proving	190
6.3.1	Parallelism at the Term or Literal Level	191
6.3.1.1	Parallelism at the Literal Level for Subgoal-Reduction Strategies	191
6.3.1.2	Parallelism at the Term Level for Ordering-Based Strategies	191
6.3.2	Parallelism at the Clause Level	193
6.3.2.1	Parallelism at the Clause Level for Subgoal-Reduction and Instance-Based Strategies	193
6.3.2.2	Parallelism at the Clause Level for Ordering-Based Strategies	194
6.3.3	The Rise of Parallel Search	196
6.3.4	Multi-search	198
6.3.4.1	Multi-search for Subgoal-Reduction Strategies	199
6.3.4.2	Multi-search for Ordering-Based Strategies	200
6.3.5	Distributed Search	202
6.3.5.1	Distributed Search for Ordering-Based Strategies	202
6.3.5.2	The Basic Clause-Diffusion Mechanisms	203
6.3.5.3	The Subdivision of Clauses in Clause-Diffusion	204
6.3.5.4	The Subdivision of Inferences in Clause-Diffusion	205
6.3.5.5	Distributed Global Contraction, Distributed Fairness, and Distributed Proof Reconstruction	206
6.3.5.6	The Clause-Diffusion Provers	207
6.4	Discussion	211

6.4.1	Parallel Theorem Proving and Parallel Satisfiability	211
6.4.2	Parallelism and First-Order Model-Based Reasoning	215
	References	217
7	Parallel Answer Set Programming	237
	Agostino Dovier, Andrea Formisano, and Enrico Pontelli	
7.1	Introduction	237
7.2	Background	240
7.2.1	Definite Logic Programming	240
7.2.2	Normal Logic Programs and Answer Set Programming	242
7.2.3	Datalog	244
7.2.4	Alternative ASP Computation Models	244
7.2.4.1	Program Completion	244
7.2.4.2	Conflict-Driven Search	245
7.2.4.3	ASP Computation	246
7.3	Parallelizing the Grounding Phase	247
7.3.1	Introduction	247
7.3.2	Naive Parallel Grounding	248
7.3.3	Multi-level Parallel Grounding	248
7.4	Parallelizing the Inference Phase I: Parallel Datalog	252
7.5	Parallelizing the Inference Phase II: Parallel ASP	253
7.5.1	Parallelizing the Search Process	253
7.5.1.1	General Idea and Seminal Work	253
7.5.1.2	Techniques for Task Sharing	255
7.5.1.3	Scheduling and Load Balancing	258
7.5.1.4	Parallelizing Lookahead	261
7.5.2	GPU-Based Parallelism	264
7.5.2.1	GPU-Based Datalog Solving	265
7.5.2.2	GPU-Based Conflict-Driven ASP Solving	266
7.5.3	Moving Towards Large-Scale Architectures	270
7.5.3.1	The Map-Reduce Programming Model	270
7.5.3.2	Datalog and Map-Reduce	271
7.5.3.3	Towards ASP: Well-Founded Semantics and Map-Reduce	272
7.5.3.4	Other Relevant Applications of Map-Reduce	274
7.5.4	Portfolio Approaches for ASP	274
7.6	Discussion and Conclusions	276
	References	277
8	Parallel Solvers for Mixed Integer Linear Optimization	283
	Ted Ralphs, Yuji Shinano, Timo Berthold, and Thorsten Koch	
8.1	Introduction	284
8.2	Sequential Algorithms	286
8.2.1	Basic Components	286
8.2.2	Advanced Procedures	288
8.3	Parallel Algorithms	290

8.3.1	Scalability and Performance	291
8.3.1.1	Scalability	292
8.3.1.2	Performance	294
8.3.2	Properties	295
8.3.2.1	Abstraction and Integration	295
8.3.2.2	Granularity	297
8.3.2.3	Adaptivity	298
8.3.2.4	Knowledge Sharing	299
8.3.2.5	Load Balancing	299
8.3.2.6	Synchronization and Coordination	303
8.3.2.7	Determinism	304
8.3.3	Implementation	305
8.3.3.1	Platform	305
8.3.3.2	Frameworks and Solvers	308
8.3.3.3	Coordination Mechanisms	308
8.4	Software	316
8.4.1	Solvers	317
8.4.2	Frameworks	321
8.5	Performance Measurement	324
8.5.1	Performance Variability	325
8.5.2	Comparisons	326
8.5.3	Instance Selection	327
8.5.4	Alternative Performance Measures	328
8.5.5	Summary Measures	328
8.6	Concluding Remarks	329
	References	329
9	Parallel Constraint Programming	337
	Jean-Charles Régin and Arnaud Malapert	
9.1	Introduction	337
9.1.1	Filtering + Propagation	339
9.1.2	Search	339
9.1.2.1	Search Methods in Solvers	341
9.1.3	Parallelism and Constraint Programming	342
9.1.3.1	Parallel Propagators and Propagation	342
9.1.3.2	Search Space Splitting	342
9.1.3.3	Portfolio Algorithms	344
9.1.3.4	Distributed CSPs	345
9.1.3.5	Problem Decomposition	345
9.2	Background	346
9.2.1	Parallelism	346
9.2.1.1	Parallelization Measures and Amdahl's Law	346
9.2.2	Embarrassingly Parallel Computation	347
9.2.3	Internal and External Parallelization	348
9.2.4	Constraint Programming	349

9.3	Parallel Search Tree	350
9.3.1	Static Partitioning	350
9.3.2	Dynamic Partitioning	351
9.3.2.1	Local Subtree Solving	352
9.3.2.2	Subtree Definition	353
9.4	Problem Decomposition	356
9.4.1	Principles	356
9.4.1.1	Sub-problems Generation: a Top-Down Method	358
9.4.1.2	Sub-problems Generation: a Bottom-Up Method	361
9.4.1.3	Implementation	362
9.4.1.4	Size of the Partition	363
9.4.2	Determinism	364
9.5	Comparison Between the Work-Stealing Approach and EPS	364
9.6	Experiments	365
9.6.1	Benchmark Instances	365
9.6.1.1	Implementation Details	366
9.6.1.2	Execution Environments	366
9.6.2	Multi-core	367
9.6.3	Data Center	368
9.6.4	Cloud Computing	369
9.6.5	Comparison with Portfolios	370
9.7	Conclusion	371
	References	372
10	Parallel Local Search	381
	Philippe Codognet, Danny Munera, Daniel Diaz, and Salvador Abreu	
10.1	Introduction	381
10.2	Local Search Metaheuristics	383
10.3	Sources of Parallelism	386
10.3.1	Single-Walk and Multiple-Walk Methods	386
10.3.2	Parallel Speedups and Runtime Distributions	387
10.4	Single-Walk Approaches	389
10.5	Independent Multiple-Walk Approaches	390
10.5.1	Early Independent Multiple-Walk Methods	390
10.5.2	Recent Experiments and Performance Results	392
10.6	Cooperative Multiple-Walk Approaches	394
10.6.1	Metaheuristic Parallelization Approaches	395
10.6.2	Agent-Based Approaches	398
10.6.3	Framework Approaches	401
10.7	Parallelism at Work	403
10.7.1	Stable Matching Problem	403
10.7.2	The Quadratic Assignment Problem	405
10.8	Conclusion	408
	References	409

11 Parallel A* for State-Space Search	419
Alex Fukunaga, Adi Botea, Yuu Jinnai, and Akihiro Kishimoto	
11.1 Introduction	419
11.2 Preliminaries: Review of A*	421
11.2.1 The A* Algorithm	423
11.3 Parallel Best-First Search Algorithms	424
11.3.1 Parallel Overheads	425
11.3.2 Centralized Parallel A*	425
11.3.3 Decentralized Parallel A*	428
11.3.3.1 Termination Detection in Decentralized Parallel Search	429
11.4 Hash-Based Decentralized A*	430
11.4.1 Hash Distributed A*	431
11.5 Decentralized Search Using Structure-Based Search Space Partitioning)	432
11.6 Hash Functions for Hash-Based Decentralized Work Distribution	433
11.6.1 Multiplicative Hashing	433
11.6.2 Zobrist Hashing	434
11.6.3 Operator-Based Zobrist Hashing	434
11.6.4 Abstraction	435
11.6.5 Abstract Zobrist Hashing	436
11.6.6 Hyperplane Work Distribution	437
11.6.7 Empirical Comparison of Hash Functions	439
11.6.8 Domain-Independent, Automatic Generation of Hash Functions	441
11.6.9 Hash-Based Work Distribution in Model Checking	441
11.7 Parallel Portfolios Using A*	442
11.8 Parallel, Limited-Memory A* (Parallel IDA*, TDS, PRA*)	443
11.8.1 Transposition Table-Driven Scheduling (TDS)	444
11.8.2 Work Stealing for IDA*	444
11.8.3 Parallel Window Search	445
11.8.4 Parallel Retracting A* (PRA*)	446
11.9 Parallel A* in Cloud Environments with Practically Unlimited Available Resources	446
11.9.1 Iterative Allocation Strategy	447
11.10 Parallel A* and IDA* on Graphics Processing Units	448
11.11 Other Approaches	449
References	450
12 Parallel Model Checking Algorithms for Linear-Time Temporal Logic	457
Jiri Barnat, Vincent Bloemen, Alexandre Duret-Lutz, Alfons Laarman, Laure Petrucci, Jaco van de Pol, and Etienne Renault	
12.1 Introduction	458
12.2 Preliminaries: LTL Model Checking and Automata	461

12.2.1 Automata-Theoretic Model Checking	461
12.2.2 Sequences and ω -Words	461
12.2.3 Linear-Time Temporal Logic	462
12.2.4 Kripke Structures	463
12.2.5 Büchi Automata	463
12.2.6 The Emptiness-Check Problem	466
12.2.7 Implicit Models and Automata	469
12.2.8 Simpler Subclasses	471
12.3 Basic Sequential LTL Model Checking Algorithms	473
12.3.1 On-the-Fly Algorithms	473
12.3.2 Depth-First Search	474
12.3.3 Nested-DFS	476
12.3.4 Algorithms Based on SCC Decomposition	478
12.4 Multi-core, DFS-Based Solutions	481
12.4.1 Terminal and Weak Acceptance	481
12.4.2 CNDFS	484
12.4.3 Multi-core/DFS-Based SCC Decomposition	486
12.5 Distributed, BFS-Based Solutions	492
12.5.1 One-Way-Catch-Them-Young	492
12.5.2 MAP	494
12.5.3 Combining OWCTY and MAP	497
12.6 Conclusion	498
References	499
13 Multi-core Decision Diagrams	509
Tom van Dijk and Jaco van de Pol	
13.1 Introduction	509
13.2 Preliminaries	510
13.2.1 Boolean Logic and Notation	511
13.2.2 Binary Decision Diagrams	511
13.2.3 Multi-terminal Binary Decision Diagrams	513
13.2.4 Algorithms on Decision Diagrams	514
13.2.5 Parallelism	516
13.2.6 Historical Perspective	517
13.3 Parallel Decision Diagrams	519
13.3.1 Work-Stealing	519
13.3.2 Parallel Operations with Work-Stealing	522
13.3.3 Conclusion	525
13.4 Concurrent Data Structures	525
13.4.1 Representation of Nodes	525
13.4.2 Unique Table	526
13.4.3 Computed Table	531
13.5 Garbage Collection	533
13.6 Empirical Results	535
13.6.1 Symbolic Model Checking	536

Part II Tools and Applications

13.6.2	Symbolic On-the-Fly Reachability	536
13.6.3	Symbolic Bimulation Minimisation	537
13.6.4	Probabilistic Model Checking	540
13.7	Conclusions	540
	References	541
14	Parallel Model-Based Diagnosis	547
	Kostyantyn Sheketykhin, Dietmar Jannach, and Thomas Schmitz	
14.1	Introduction	547
14.1.1	Background	547
14.1.2	Outline of the Chapter	548
14.2	Reiter's Diagnosis Framework	549
14.2.1	Example: A Diagnosis Problem Instance	549
14.2.2	Diagnoses and Conflicts	551
14.2.3	The Hitting Set Tree Algorithm	553
14.2.4	Example: Hitting Set Tree Construction	554
14.2.5	Complexity Considerations	556
14.3	Alternative Approaches to Compute Diagnoses	556
14.4	Parallelization of Tree Search Algorithms	559
14.4.1	General Parallelization Strategies	559
14.4.2	Applying Domain-Independent Parallelized Search Techniques	561
14.5	Parallelized Hitting Set Tree Construction Schemes	562
14.5.1	Computing Multiple Hitting Set Tree Nodes in Parallel	562
14.5.1.1	Level-Wise Parallelization	563
14.5.1.2	Full Parallelization	564
14.5.2	Computing Nodes and Conflicts in Parallel	565
14.5.2.1	Background: QUICKXPLAIN and MERGEXPLAIN	566
14.5.2.2	Strategies for Combining Node and Conflict Computation	567
14.6	Effectiveness of Computing Multiple Nodes in Parallel	569
14.6.1	General Considerations	569
14.6.2	Results for Standard Electronic Circuit Benchmark Problems	571
14.6.3	Systematic Variation of Problem Characteristics	572
14.6.3.1	Method	572
14.6.3.2	Results	574
14.7	Alternative Model-Based Diagnosis Parallelization Approaches	574
14.7.1	Tree-Based Approaches To Find One or Few Diagnoses	574
14.7.2	Distributed Hitting Set Algorithms with Known Conflicts	576
14.8	Summary	577
	References	578

15	Selection and Configuration of Parallel Portfolios	583
	Marius Lindauer, Holger Hoos, Frank Hutter, and Kevin Leyton-Brown	
15.1	Introduction	584
15.2	Per-Instance Selection of Parallel Portfolios	586
15.2.1	Problem Statement	586
15.2.2	Parallelization of Sequential Algorithm Selectors	588
15.2.2.1	Performance-Based Nearest Neighbor (PNN)	589
15.2.2.2	Distance-Based Nearest Neighbor (DNN)	589
15.2.2.3	Clustering	590
15.2.2.4	Regression	590
15.2.2.5	Pairwise Voting	591
15.2.3	Parallel Presolving Schedules	591
15.2.4	Empirical Study on Satisfiability Benchmarks	591
15.2.5	Other Parallel Portfolio Selection Approaches	594
15.3	Automatic Construction of Parallel Portfolios from Parameterized Solvers	595
15.3.1	Problem Statement	596
15.3.2	Automatic Construction of Parallel Portfolios (ACPP)	598
15.3.2.1	Multiplying Configuration Space: GLOBAL	599
15.3.2.2	Iterative Approach: PARHYDRA	599
15.3.2.3	Comparing GLOBAL and PARHYDRA	601
15.3.2.4	Empirical Study on SAT 2012 Challenge	602
15.3.2.5	ACPP with Multiple Solvers	602
15.3.3	Automatic Construction of Parallel Portfolios from Parallel Parameterized Solvers	603
15.3.3.1	Configuration of Clause Sharing	603
15.3.3.2	Portfolio Construction Using Parallel Solvers	604
15.3.3.3	Empirical Study on 2012 SAT Challenge	606
15.4	Conclusions and Future Work	607
	References	609
16	An Application of Parallel Satisfiability Solving to the Verification of Complex Embedded Systems	617
	Orlando Ferrante, Alberto Ferrari, Christos Sofronis, Leonardo Mangeruca, and Luca Benvenuti	
16.1	Introduction	617
16.2	FormalSpecs Verifier Verification Framework	618
16.3	Integration of the ManySAT Solver	619
16.4	Cruise Control Use Case	620
16.5	Simulink Model and Specification	622
16.5.1	Continuous-Time Non-linear Model	622
16.5.1.1	ECU Subsystem	622
16.5.1.2	Engine Subsystem	623

16.5.1.3	Vehicle Dynamics Subsystem	624
16.5.2	Discrete-Time Discrete-Value Model	625
16.5.2.1	Verification Subsystems	627
16.6	Experimental Results	628
16.6.1	Cruise Control Model	628
16.6.1.1	Bounded Model Checking Verification with Incremental Bounds	629
16.6.1.2	Bounded Model Checking Verification with Fixed Bound Value	629
16.6.2	Additional Experiments	630
16.7	Conclusions	631
	References	631
17	Parallel Constraint-Based Local Search: An Application to Designing Resilient Long-Reach Passive Optical Networks	633
	Alejandro Arbelaez, Deepak Mehta, Barry O'Sullivan, and Luis Quesada	
17.1	Introduction	634
17.2	Formal Specification and Complexity	635
17.3	A Mathematical Model for ERDCMST	637
17.4	Iterated Constraint-Based Local Search	638
17.4.1	Move Operators	639
17.4.2	Operations and Complexities	640
17.5	Sequential Algorithm	644
17.6	Parallel Algorithm	646
17.6.1	Multi-Walk and Single-Walk	646
17.6.2	Parallel Moves for ERDCMST	647
17.7	Application: Long-Reach Passive Optical Networks	650
17.8	Empirical Evaluation	652
17.8.1	ERDCMST Results: Sequential LS	653
17.8.2	ERDCMST Results: Parallel LS	656
17.9	Conclusions and Future Work	662
	References	662
	List of Algorithms	669
	Index	671