

Contents

<i>Preface</i>	page xi
<i>Acknowledgments</i>	xiv
1 What Is a Kernel?	1
1.1 Introduction	1
1.2 Kernelization: Formal Definition	6
PART I UPPER BOUNDS	13
2 Warm Up	15
2.1 Trivial Kernelization	16
2.2 VERTEX COVER	18
2.3 FEEDBACK ARC SET IN TOURNAMENTS	21
2.4 DOMINATING SET in Graphs of Girth at Least 5	22
2.5 Alternative Parameterization for VERTEX COVER	25
2.6 EDGE CLIQUE COVER	28
3 Inductive Priorities	32
3.1 Priorities for MAX LEAF SUBTREE	33
3.2 Priorities for FEEDBACK VERTEX SET	40
4 Crown Decomposition	50
4.1 Crown Decomposition	51
4.2 VERTEX COVER and DUAL COLORING	52
4.3 MAXIMUM SATISFIABILITY	55
4.4 LONGEST CYCLE Parameterized by Vertex Cover	57

5	Expansion Lemma	61
5.1	Expansion Lemma	61
5.2	CLUSTER VERTEX DELETION: Bounding the Number of Cliques	65
5.3	Weighted Expansion Lemma	66
5.4	Component Order Connectivity	70
5.5	FEEDBACK VERTEX SET	73
6	Linear Programming	84
6.1	The Theorem of Nemhauser and Trotter	84
6.2	2-SAT of Minimum Weight	89
6.3	Reduction of MIN-WEIGHT-2-IP to MIN-ONES-2-SAT	92
6.4	COMPONENT ORDER CONNECTIVITY	96
7	Hypertrees	105
7.1	Hypertrees and Partition-Connectedness	105
7.2	SET SPLITTING	108
7.3	MAX-INTERNAL SPANNING TREE	114
8	Sunflower Lemma	121
8.1	Sunflower Lemma	121
8.2	d -HITTING SET	122
8.3	d -SET PACKING	123
8.4	Domination in Degenerate Graphs	124
8.5	Domination in K_{ij} -Free Graphs	128
9	Modules	133
9.1	Modular Partition	133
9.2	CLUSTER EDITING	139
9.3	COGRAPH COMPLETION	147
9.4	FAST Revisited	158
10	Matroids	164
10.1	Matroid Basics	164
10.2	Cut-Flow Data Structure	169
10.3	Kernel for ODD CYCLE TRANSVERSAL	173
11	Representative Families	183
11.1	Introduction to Representative Sets	183
11.2	Computing Representative Families	185
11.3	Kernel for VERTEX COVER	191
11.4	DIGRAPH PAIR CUT	192

11.5	An Abstraction	199
11.6	Combinatorial Approach	202
12	Greedy Packing	217
12.1	SET COVER	218
12.2	MAX-LIN-2 above Average	222
12.3	MAX-ER-SAT	231
13	Euler's Formula	237
13.1	Preliminaries on Planar Graphs	237
13.2	Simple Planar Kernels	238
13.3	Planar FEEDBACK VERTEX SET	243
	PART II META THEOREMS	255
14	Introduction to Treewidth	257
14.1	Properties of Tree Decompositions	259
14.2	Computing Treewidth	262
14.3	Nice Tree Decompositions	265
14.4	Dynamic Programming	268
14.5	Treewidth and MSO_2	279
14.6	Obstructions to Bounded Treewidth	286
15	Bidimensionality and Protrusions	297
15.1	Bidimensional Problems	298
15.2	Separability and Treewidth Modulators	301
15.3	Protrusion Decompositions	306
15.4	Kernel for DOMINATING SET on Planar Graphs	309
16	Surgery on Graphs	316
16.1	Boundaried Graphs and Finite Integer Index	319
16.2	Which Problems Have Finite Integer Index?	323
16.3	A General Reduction Rule	327
16.4	Kernelization in Quadratic Running Time	333
16.5	Linear Time Algorithm	340
	PART III LOWER BOUNDS	357
17	Framework	359
17.1	OR-Distillation	360
17.2	Cross-Composition	366
17.3	Examples of Compositions	369

18	Instance Selectors	377
18.1	DISJOINT FACTORS	379
18.2	SAT Parameterized by the Number of Variables	381
18.3	Colored Red-Blue Dominating Set	383
19	Polynomial Parameter Transformation	389
19.1	Packing Paths and Cycles	390
19.2	RED-BLUE DOMINATING SET	392
20	Polynomial Lower Bounds	398
20.1	Weak Cross-Composition	398
20.2	Lower Bound for VERTEX COVER	401
20.3	Lower Bound for d -HITTING SET	404
20.4	RAMSEY	408
21	Extending Distillation	412
21.1	Oracle Communication Protocol	412
21.2	Hardness of Communication	414
21.3	Lower Bounds for POINT LINE COVER	418
21.4	Lower Bounds Using Co-Nondeterminism	424
21.5	AND-Distillations and AND-Compositions	425
	PART IV BEYOND KERNELIZATION	427
22	Turing Kernelization	429
22.1	MAX LEAF SUBTREE	431
22.2	PLANAR LONGEST CYCLE	431
23	Lossy Kernelization	440
23.1	Framework	441
23.2	Cycle Packing	455
23.3	Partial Vertex Cover	457
23.4	Connected Vertex Cover	458
23.5	Steiner Tree	461
Appendix A	Open Problems	467
A.1	Polynomial Kernels	467
A.2	Structural Kernelization Bounds	469
A.3	Deterministic Kernels	471
A.4	Turing Kernels	472

Appendix B	Graphs and SAT Notation	474
Appendix C	Problem Definitions	477
	<i>References</i>	483
	<i>Author Index</i>	505
	<i>Index</i>	510

Preprocessing, also known as *rule reduction*, is one of the basic and most natural types of heuristic algorithms. The idea of preprocessing information to speed up computations can be traced much before the invention of the first computers. The book *Mirjehi Logarithmarum Canonis Descriptio* (A Description of the Admirable Table of Logarithms) authored by Napier (1550–1617), who is credited with the invention of logarithms, was published in 1614. A quote (Giles 1969) attributed to Laplace states that this table, “by shortening the labours, doubled the life of the astronomer.”

As a strategy of coping with hard problems, preprocessing is universally employed. A recent survey of preprocessing, ranging from lossless data compression and navigation systems to secondary data analysis for the classification of cancer types. The “gold standard” successes in software development for hard problems, such as CPLEX for integer linear programming, depend heavily on sophisticated preprocessing routines. However, even very simple preprocessing can be surprisingly effective.

Developing rigorous mathematical theories that explain the behavior of practical algorithms and heuristics has become an increasingly important challenge in the theory of computing for the twenty-first century. Addressing this central in the theory of computing, the report of Condon et al. (1999) states:

While theoretical work on models of computation and methods for analyzing algorithms has had enormous payoff, we are not done. In many situations, simple algorithms do well. We don’t understand why! It is apparent that worst case analysis does not provide useful insights on the performance of algorithms and heuristics and our models of computation need to be further developed and refined.”