

Commenced Publication in 1973

Founding and Former Series Editors:

Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

David Hutchison

Lancaster University, UK

Takeo Kanade

Carnegie Mellon University, Pittsburgh, PA, USA

Josef Kittler

University of Surrey, Guildford, UK

Jon M. Kleinberg

Cornell University, Ithaca, NY, USA

Alfred Kobsa

University of California, Irvine, CA, USA

Friedemann Mattern

ETH Zurich, Switzerland

John C. Mitchell

Stanford University, CA, USA

Moni Naor

Weizmann Institute of Science, Rehovot, Israel

Oscar Nierstrasz

University of Bern, Switzerland

C. Pandu Rangan

Indian Institute of Technology, Madras, India

Bernhard Steffen

TU Dortmund University, Germany

Madhu Sudan

Microsoft Research, Cambridge, MA, USA

Demetri Terzopoulos

University of California, Los Angeles, CA, USA

Doug Tygar

University of California, Berkeley, CA, USA

Gerhard Weikum

Max Planck Institute for Informatics, Saarbruecken, Germany

Hans L. Bodlaender Rod Downey
Fedor V. Fomin Dániel Marx (Eds.)

The Multivariate Algorithmic Revolution and Beyond

Essays Dedicated to Michael R. Fellows
on the Occasion of His 60th Birthday

Volume Editors

Hans L. Bodlaender

Utrecht University, Department of Information and Computing Sciences

P.O. Box 80.089, 3508 TB Utrecht, The Netherlands

E-mail: h.l.bodlaender@uu.nl

Rod Downey

Victoria University, School of Mathematics, Statistics and Operations Research

P.O. Box 600, Wellington, New Zealand

E-mail: rod.downey@vuw.ac.nz

Fedor V. Fomin

University of Bergen, Institute of Informatics

Postboks 7803, 5020 Bergen, Norway

E-mail: fomin@ii.uib.no

Dániel Marx

Hungarian Academy of Sciences (MTA SZTAKI)

Computer and Automation Research Institute

Pf. 63, 1518 Budapest, Hungary

E-mail: dmarx@cs.bme.hu

ISSN 0302-9743

e-ISSN 1611-3349

ISBN 978-3-642-30890-1

e-ISBN 978-3-642-30891-8

DOI 10.1007/978-3-642-30891-8

Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2012938949

CR Subject Classification (1998): F.2, G.2, I.3.5, E.1, F.1, F.4.1

LNCS Sublibrary: SL 1 – Theoretical Computer Science and General Issues

© Springer-Verlag Berlin Heidelberg 2012

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.

The use of general descriptive names, registered names, trademarks, 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.

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

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



Michael R. Fellows

Preface

A Festschrift allows the scientific community to acknowledge the contributions of a scientist turning 60. In the case of Mike Fellows, we have much to acknowledge. Mike has made crucial contributions involving fundamental paradigm shifts in computer science and in mathematics/computing education.

Many researchers can make a technical contribution to science, but only a few can change the way we understand the world. Quite aside from the intrinsic difficulty of being able to visualize such a change in our thinking, when a paradigm shift is involved, there is often strong resistance to the new ideas. The initiators need vision and a strength of will to carry the program through.

Building upon early work with Mike Langston, Mike Fellows and Rod Downey founded the field of parameterized complexity. This is an approach toward understanding the complexity of computations which occur in practice. It seeks to use a multivariate approach to understand the exact contribution of each part of a problem in its computational complexity.

This is now a thriving field in computational complexity. Sensitizing algorithm designers to this paradigm has allowed for a distinctive set of tools to be developed. These tools allow for a systematic and extended dialog with a problem. This is a field consciously trying to tie together theoretical computer science and applications in a meaningful way, and has grown exponentially in the last 15 or so years.

The majority of contributions in this volume are in the area of parameterized complexity, with surveys from leading experts, including a basic guide to the area, and personal memories by those involved in the development.

It is very unusual for a world-class researcher in science to *also* be involved in education, and this seems especially true of *computer* science. With Nancy Casey, Tim Bell, Neil Koblitiz and others, Mike began the remarkable project that became *Computer Science Unplugged*. This was another paradigm shift, this time the initiative was in computing and mathematics education. The idea was to involve young children with insights from advanced computer science. One of the excellent contributions to this volume is the article describing what this is all about, how it came to be, and the trials and tribulations of getting the ideas to be accepted.

Mike Fellows is a remarkable scientist. Not only has he made deep and lasting contributions to human knowledge, he has been instrumental in the creation of connected research communities throughout the world. He lectures, goes to primary schools, interacts everywhere and is extremely generous with his ideas. This theme comes through in the reminiscences in the present volume.

Turning 60 was traditionally a time when people slowed down, perhaps looking back fondly on their lives and preparing for retirement. Nothing seems further from the truth in Mike's case. Mike and Fran now spend their days in hectic

trans-world trips from Australia to India to Europe and the US. Mike is now involved in more papers per year than when he was in his twenties, and keeps a schedule that would pole-axe most people. To which we say: long may it last.

We thank a number of people that made this volume possible: Alex Downey for artwork, all authors, all anonymous referees and proofreaders, Saket Saurabh for the initial idea for the Festschrift, Fran Rosamond for help in many ways, and Anna Kramer, Ronen Nugent and the other people at Springer for helping make this possible and being enthusiastic about the project. This Festschrift contains some superb surveys and fascinating insights into a prominent scientist. Enjoy.

April 2012

Hans Bodlaender
Rod Downey
Fedor Fomin
Dániel Marx

Curriculum Vitae Michael R. Fellows

Current position

Professor (Australian Professorial Fellow)

Address

Charles Darwin University
Darwin, Northern Territory, Australia
phone: (international mobile) +44 7590 089314
electronic mail: michael.fellows@cdu.edu.au
fax: +61 8 8946 6680

Personal information

Born: June 15, 1952, in Upland, California
Married: to Frances Rosamond (Professor, Charles Darwin University)
Two children

Education

Ph.D., Computer Science, University of California, San Diego, 1985
M.A., Mathematics, University of California, San Diego, 1982
B.A., Mathematics, Sonoma State University, California, 1980

Experience

2010–present: Professor, Charles Darwin University, Australia
2001–2010: Professor of Computer Science, University of Newcastle,
Australia
1999–2001: Reader of Theoretical Computer Science, Victoria University,
New Zealand
1995–2001: Professor of Computer Science, University of Victoria, Canada
1990–1995: Associate Professor, University of Victoria, Canada
1987–1990: Associate Professor, University of Idaho, USA
1986–1987: Assistant Professor, University of New Mexico, USA
1985–1986: Assistant Professor, Washington State University, USA

Professional Interests

Computational Complexity Theory, Combinatorial Algorithms,
Computational Biology, Mathematical Sciences Communication

Refereed Journal Publications

1. "A Topological Parameterization and Hard Graph Problems," *Congressus Numerantium* 59 (1987), 69–78, with F. Hickling and M. Syslo.
2. "Computational Complexity of Integrality," *Journal of Combinatorial Mathematics and Combinatorial Computing* 2 (1987), 179–191, with L. H. Clark and R. C. Entringer.
3. "Nonconstructive Proofs of Polynomial-Time Complexity," *Information Processing Letters* 26(1987/88), 157–162, with M. A. Langston.
4. "Processor Utilization in a Linearly Connected Parallel Processing System," *IEEE Transactions on Computers* 37 (1988), 594–603, with M. A. Langston.
5. "On Finding Optimal and Near-Optimal Lineal Spanning Trees," *Algorithmica* 3 (1988), 549–560, with D. K. Friesen and M. A. Langston.
6. "Nonconstructive Tools for Proving Polynomial-Time Complexity," *Journal of the Association for Computing Machinery* 35 (1988) 727–739, with M. A. Langston.
7. "On the Galactic Number of a Hypercube," *Mathematical and Computer Modelling* 11 (1988), 212–215, with M. Hoover and F. Harary.
8. "Radius and Diameter in Manhattan Lattices," *Discrete Mathematics* 73 (1989), 119–125, with D. J. Kleitman.
9. "Polynomial-Time Self-Reducibility: Theoretical Motivations and Practical Results," *International Journal of Computer Mathematics* 31 (1989), 1–9, with D. J. Brown and M. A. Langston.
10. "The Robertson-Seymour Theorems: A Survey of Applications," *Contemporary Mathematics* 89 (1989), 1–18.
11. "Counting Spanning Trees in Directed Regular Multigraphs," *Journal of the Franklin Institute* 326 (1989), 889–896, with J. M. Wojciechowski.
12. "The Immersion Order, Forbidden Subgraphs and the Complexity of Network Integrality," *Journal of Combinatorial Mathematics and Combinatorial Computing* 6 (1989), 23–32, with S. Stueckle.
13. "Transversals of Vertex Partitions in Graphs," *SIAM J. Discrete Math.* 3 (1990), 206–215.
14. "Searching for $K_{3,3}$ in Linear Time," *Linear and Multilinear Algebra* 29 (1991), 279–290, with P. A. Kaschube.
15. "Perfect Domination," *Australasian J. Combinatorics* 3 (1991), 141–150, with M. Hoover.
16. "Fast Search Algorithms for Graph Layout Permutation Problems," *Integration, the VLSI Journal* 12 (1991), 321–337, with M. A. Langston.
17. "Cycles of Length 0 Modulo 3 in Graphs," *Annals of Discrete Math.* (1991), 87–101, with C. A. Barefoot, L. H. Clark, J. Douthett and R. C. Entringer.
18. "Constructive Complexity," *Discrete Applied Math.* 34 (1991), 3–16, with K. Abrahamson, M. A. Langston and B. Moret. (Also published in the book series *Annals of Discrete Mathematics*, in: *Combinatorics and Theoretical Computer Science*, R. Simion, ed., North-Holland, 1992, pp. 3–16.)

19. "On the Complexity and Combinatorics of Covering Finite Complexes," *Australasian J. Combinatorics* 4 (1991), 103-112, with J. Abello and J. Stillwell.
20. "On Well-Partial-Order Theory and Its Application to Combinatorial Problems of VLSI Design," *SIAM J. Discrete Math.* 5 (1992), 117-126, with M. A. Langston.
21. "Small Diameter Symmetric Networks From Linear Groups," *IEEE Transactions on Computers* 40 (1992), 218-220, with L. Campbell, G. E. Carlsson, M. J. Dinneen, V. Faber, M. A. Langston, J. W. Moore, A. P. Mullhaupt and H. B. Sexton.
22. "Fixed-Parameter Tractability and Completeness," *Congressus Numerantium* 87 (1992), 161-178, with R. G. Downey.
23. "Self-Witnessing Polynomial-Time Complexity and Certificates for Primality," *Designs, Codes and Cryptography* 2 (1992), 231-235, with N. Koblitz.
24. "The Private Neighborhood Cube," *SIAM J. Discrete Math.* 7 (1994), 41-47, with G. Fricke, S. Hedetniemi and D. Jacobs.
25. "Cultural Aspects of Mathematics Education Reform," *Notices of the American Mathematics Society* 41 (1994), 5-9, with A. Hibner and N. Koblitz.
26. "On Search, Decision and the Efficiency of Polynomial-Time Algorithms," *Journal of Computer and Systems Science* 49 (1994), 769-779, with M. A. Langston.
27. "The Complexity of Induced Minors and Related Problems," *Algorithmica* 13 (1995), 266-282, with J. Kratochvíl, M. Middendorf and F. Pfeiffer.
28. "Large Planar Graphs with Given Diameter and Maximum Degree," *Discrete Applied Math.* 61 (1995), 133-153, with P. Hell and K. Seyffarth.
29. "Fixed-Parameter Tractability and Completeness I: Basic Theory," *SIAM J. Computing* 24 (1995), 873-921, with R. Downey.
30. "Fixed-Parameter Tractability and Completeness II: Completeness for $W[1]$," *Theoretical Computer Science A* 141 (1995), 109-131, with R. Downey.
31. "Fixed Parameter Tractability and Completeness IV: On Completeness for $W[P]$ and PSPACE Analogs," *Annals of Pure and Applied Logic* 73 (1995), 235-276, with K. Abrahamson and R. Downey.
32. "The Parameterized Complexity of the Longest Common Subsequence Problem," *Theoretical Computer Science A* 147 (1995), 31-54, with H. Bodlaender, R. Downey and H.T. Wareham.
33. "Parameterized Complexity Analysis in Computational Biology," *Computer Applications in the Biosciences* 11 (1995), 49-57, with H. Bodlaender, R. Downey, M. Hallett, and H.T. Wareham.
34. "On the Parameterized Complexity of Problems in NP," *Information and Computation* 123 (1995), 38-49, with L. Cai, J. Chen and R. Downey.
35. "On the Complexity of k -Processor Scheduling," *Operations Research Letters* 18 (1995), 93-98, with H. Bodlaender.
36. "Vertex Transversals That Dominate," *Journal of Graph Theory* 21 (1996), 21-32, with N. Alon and D.O. Hare.
37. "A Simple Linear Time Algorithm for Finding Path Decompositions of Small Width," *Information Processing Letters* 57 (1996), 197-203, with K. Cattell and M. J. Dinneen.

38. "Sparse Parameterized Problems," *Annals of Pure and Applied Logic* 82 (1996), 1–15, with M. Cesati.
39. "Approaches to Detection of Distantly Related Proteins by Database Searches," *BioTechniques* 21 (1996), 1118–1125, with K. Cattell, R. Olafson, B. Koop, I. Bailey, R.W. Olafson and C. Upton.
40. "Advice Classes of Parameterized Tractability," *Annals of Pure and Applied Logic* 84 (1997), 119–138, with L. Cai, J. Chen and R.G. Downey.
41. "The Parameterized Complexity of Short Computation and Factorization," Proceedings of the *Sacks Symposium*, in *Archive for Mathematical Logic* 36 (1997), 321–338, with L. Cai, J. Chen and R. Downey.
42. "A Note on the Computability of Obstruction Sets for Monadic Second Order Ideals," *Journal of Universal Computer Science* 3 (1997), 1194–1198, with B. Courcelle and R. Downey.
43. "Parameterized Circuit Complexity and the W Hierarchy," *Theoretical Computer Science A* 191 (1998), 97–115, with R. G. Downey and K. W. Regan.
44. "An Improved Fixed-Parameter Algorithm for Vertex Cover," *Information Processing Letters* 65 (1998), 163–168, with R. Balasubramanian and V. Raman.
45. "Constructions of Dense Planar Networks," *Networks* 32 (1998), 275–281, with P. Hell and K. Seyffarth.
46. "Threshold Dominating Sets and An Improved Characterization of $W[2]$," *Theoretical Computer Science A* 209 (1998), 123–140, with R. G. Downey.
47. "The Parameterized Complexity of Some Fundamental Problems in Coding Theory," *SIAM J. Computing* 29 (1999), 545–570, with R.G. Downey, A. Vardy and G. Whittle.
48. "On Computing Graph Minor Obstruction Sets," *Theoretical Computer Science A* 233 (2000), 107–127, with K. Cattell, M.J. Dinneen, R.G. Downey and M.A. Langston.
49. "The Complexity of Irredundant Sets Parameterized by Size," *Discrete Applied Math.* 100 (2000), 155–167, with R.G. Downey and V. Raman.
50. "The Hardness of Perfect Phylogeny, Feasible Register Assignment and Other Problems on Thin Colored Graphs," *Theoretical Computer Science A* 244 (2000), 167–188, with H. Bodlaender, M. Hallett, H. Wareham and T. Warnow.
51. "Index Sets and Parametric Reductions," *Archive for Mathematical Logic* 40 (2001), 329–348, with R.G. Downey.
52. "Forbidden Minors to Graphs with Small Feedback Sets," *Discrete Mathematics* 230 (2001), 215–252, with K. Cattell and M. Dinneen.
53. "On the Parameterized Complexity of Minimizing Tardy Tasks," *Theoretical Computer Science A* 298 (2003), 317–324, with C. McCartin.
54. "Analogues and Duals of the MAST Problem for Sequences and Trees," *Journal of Algorithms* 49 (2003), 192–216, with M. Hallett and U. Stege.
55. "Explaining Cryptographic Ideas to the General Public," *Computers and Education* 40 (2003), 199–215, with T. Bell, I. Witten and N. Koblitz.
56. "Foreword from the Guest Editors," *Journal of Computer and Systems Science* 67 (2003), 653–654, with J. Chen.

57. “Cutting Up is Hard to Do: the Parameterized Complexity of k -Cut and Related Problems,” *Electronic Notes in Theoretical Computer Science* 78 (2003), 205–218, with R. Downey, V. Estivill-Castro, E. Prieto-Rodriguez and F. Rosamond.
58. “Polynomial-Time Data Reduction for Dominating Set,” *Journal of the ACM* 51 (2004), 363–384, with J. Alber and R. Niedermeier.
59. “The Dominating Set Problem is Fixed Parameter Tractable on Graphs of Bounded Genus,” *Journal of Algorithms* 52 (2004), 152–168, with H. Fan and J. Ellis.
60. “Refined Search Tree Technique for Dominating Sets on Planar Graphs,” *Journal of Computer and Systems Science* 71 (2005), 385–405, with J. Alber, H. Fan, H. Fernau, R. Niedermeier, F. Rosamond and U. Stege.
61. “Tight Lower Bounds for Certain Parameterized NP-Hard Problems,” *Information and Computation* 201 (2005), 216–231, with J. Chen, B. Chor, X. Huang, D. Juedes, I. Kanj and G. Xia.
62. “On Finding Short Resolution Refutations and Small Unsatisfiable Subsets,” *Theoretical Computer Science* 351 (2006), 351–359, with S. Szeider and G. Wrightson.
63. “On the Parameterized Intractability of Motif Search Problems,” *Combinatorica* 26 (2006), 141–167, with J. Gramm and R. Niedermeier.
64. “A Fixed-Parameter Approach to Two-Layer Planarization,” *Algorithmica* 45 (2006), 159–182, with V. Dujmovic, M. Hallett, M. Kitching, G. Liotta, C. McCartin, N. Nishimura, P. Ragde, F. Rosamond, M. Suderman, S. Whitesides and D. R. Wood.
65. “An $O(2^{O(k)})$ FPT Algorithm for the Undirected Feedback Vertex Set Problem,” *Theory of Computing Systems* 41 (2007), 479–492, with F. Dehne, M. Langston, F. Rosamond and K. Stevens.
66. “Crown Structures for Vertex Cover Kernelization,” *Theory of Computing Systems* 41 (2007), 411–431, with F. Abu-Khazam, M. Langston and W. Suters.
67. “On the Efficiency of Polynomial Time Approximation,” *Theory of Computing Systems* 41 (2007), 459–477, with L. Cai, D. Juedes and F. Rosamond.
68. “On the Complexity of Lobbying in Multiple Referenda,” *Review of Economic Design* 11 (2007), 217–224, with R. Christian, F. Rosamond and A. Slinko.
69. “Parameterized Approximation for Dominating Set Problems,” *Information Processing Letters* 109 (2008), 68–70, with R. Downey, C. McCartin and F. Rosamond.
70. “On the Parameterized Complexity of Layered Graph Drawing,” *Algorithmica* 52 (2008), 267–292, with V. Dujmovic, M. Kitching, G. Liotta, C. McCartin, N. Nishimura, P. Ragde, F. Rosamond, M. Suderman, S. Whitesides and D. R. Wood.
71. “The Computer Journal Special Issue on Parameterized Complexity: Foreword by the Guest Editors,” *The Computer Journal* 51(1) (2008), 1–6, with R. Downey and M. Langston.

72. “Faster Fixed-Parameter Tractable Algorithms for Matching and Packing Problems,” *Algorithmica* 52 (2008), 167–176, with C. Knauer, N. Nishimura, P. Ragde, F. Rosamond, U. Stege, D. Thilikos and S. Whitesides.
73. “Cliquewidth is NP-Complete,” *SIAM Journal on Discrete Mathematics* 23(2): 909–939 (2009), with F. Rosamond, U. Rotics and S. Szeider.
74. “Derivation of Algorithms for Cutwidth and Related Graph Layout Parameters,” *Journal of Computer and System Sciences* 75 (2009), 231–244, with H.L. Bodlaender and D.M. Thilikos.
75. “The Complexity Ecology of Parameters: An Illustration Using Bounded Max Leaf Number,” *Theory of Computing Systems* 45 (2009), 822–848, with D. Lokshtanov, N. Misra, M. Mnich, F. Rosamond and S. Saurabh.
76. “On the Fixed-Parameter Intractability and Tractability of Multiple-Interval Graph Problems,” *Theoretical Computer Science* 410 (2009), 53–61, with D. Hermelin and F. Rosamond.
77. “On Problems Without Polynomial Kernels,” *Journal of Computer and System Sciences* 75 (2009), 423–434, with H.L. Bodlaender, R. Downey and D. Hermelin.
78. “Fixed-Parameter Algorithms for Kemeny Ranking,” *Theoretical Computer Science* 410 (2009), 4554–4570, with N. Betzler, J. Guo, R. Niedermeier and F. Rosamond.
79. “Clustering with Partial Information,” *Theoretical Computer Science* 411 (2010), 1202–1211, with H.L. Bodlaender, P. Heggernes, F. Mancini, C. Papadopoulos and F. Rosamond.
80. “W-Hierarchies Defined by Symmetric Gates,” *Theory of Computing Systems* 46 (2010), 311–339, with J. Flum, D. Hermelin, M. Mueller and F. Rosamond.
81. “The Parameterized Complexity of Some Minimum Label Problems,” *Journal of Computer and System Sciences* 76 (2010), 727–740, with J. Guo and I. Kanj.
82. “Graph-Based Data Clustering with Overlaps,” *Discrete Optimization* 8 (2011), 2–17, with J. Guo, C. Komusiewicz, R. Niedermeier and J. Uhlmann.
83. “On the Complexity of Some Colorful Problems Parameterized by Tree-width,” *Information and Computation* 209 (2011), 143–153, with F. Fomin, D. Lokshtanov, F. Rosamond, S. Saurabh, S. Szeider and C. Thomassen.
84. “Facility Location Problems: A Parameterized View,” *Discrete Applied Mathematics* 159 (2011), 1118–1130, with H. Fernau.
85. “Preface: Special Issue on Parameterized Complexity of Discrete Optimization,” *Discrete Optimization* 8 (2011), 1, with F. Fomin and G. Gutin.
86. “A Generalization of Nemhauser and Trotter’s Local Optimization Theorem,” *Journal of Computer and System Sciences* 77 (2011), 1141–1158, with J. Guo, H. Moser and R. Niedermeier.
87. “Parameterized Algorithmics for Finding Connected Motifs in Biological Networks,” *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 8 (2011), 1296–1308, with N. Betzler, R. van Bevern, C. Komusiewicz and R. Niedermeier.

88. “Quadratic Kernelization for Convex Recoloring of Trees,” *Algorithmica* 61 (2011), 362–378, with H. Bodlaender, M. Langston, M. Ragan, F. Rosamond and M. Weyer.
89. “Upper and Lower Bounds for Finding Connected Motifs in Vertex-Colored Graphs,” (conference version presented at ICALP 2007), *Journal of Computer and System Sciences* 77 (2011), 799–811, with G. Fertin, D. Hermelin and S. Vialette.
90. “Haplotype Inference Constrained by Plausible Haplotype Information,” *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 8 (2011), with D. Hermelin, G. Landau, F. Rosamond, L. Rozenberg and L. Tzavika.
91. “A Complexity Dichotomy for Finding Disjoint Solutions of Vertex Deletion Problems,” *ACM Transactions on Computation Theory* 2 (2011), 5–25, with J. Guo, H. Moser and R. Niedermeier.
92. “Parameterizing by the Number of Numbers,” *Theory of Computing Systems* 50 (2012), 675–693, with S. Gaspers and F. Rosamond.
93. “Local Search: Is brute-force avoidable?” *Journal of Computer and System Sciences* 78 (2012), 707–719, with F. Fomin, D. Lokshtanov, F. Rosamond, S. Saurabh and Y. Villanger.
94. “The Parameterized Complexity of Stabbing Rectangles,” *Algorithmica* 62 (2012), 564–594, with M. Dom, F. Rosamond and S. Sikdar.
95. “Well-Quasi-Orders in Subclasses of Bounded Treewidth Graphs and their Algorithmic Applications,” *Algorithmica*, to appear, with D. Hermelin and F. Rosamond.
96. “On the Parameterized Complexity of the Discrete Milling Problem with Turn Costs,” *Journal of Discrete Algorithms*, to appear, with P. Giannopoulos, C. Knauer, C. Paul, F. Rosamond, S. Whitesides and N. Yu.
97. “Towards Full Multivariate Algorithmics: Parameter Ecology and the Deconstruction of Computational Complexity,” *European J. Combinatorics*, to appear, with B. M. P. Jansen and F. A. Rosamond.

Books

1. *This is Mega-Mathematics!*, 134 pp., available for free at the World Wide Web site: <http://www.c3.lanl.gov/~captors/mega-math>, 1992, with N. Casey.
2. *Computer Science Unplugged ... offline activities and games for all ages*, 231 pp., 1996, with T. Bell and I. Witten.
3. *Parameterized Complexity*, 530 pp., Springer-Verlag, 1999, with R.G. Downey.

Book Contributions

1. M.R. Fellows, “Parameterized complexity: new developments and research frontiers.” In R.G. Downey and D. Hirschfeldt (eds.), *Aspects of Complexity*, pp. 51–72. de Gruyter Series in Logic and Its Applications, Vol. 4, de Gruyter, Berlin, 2000.

2. M. Fellows, S. Gaspers and F. Rosamond, “Multivariate Complexity Theory.” In E.K. Blum and A.V. Aho (eds.), *Computer Science: The Hardware, Software and Heart of It*, pp. 269–294. Springer, 2011.

Refereed Conference Proceedings

1. “On Finding Obstruction Sets and Polynomial-Time Algorithms for Gate Matrix Layout,” *Proceedings of the 25th Allerton Conference on Communication, Control and Computing* (1987), 397–398, with R. L. Bryant, N. G. Kinnersley and M. A. Langston.
2. “Layout Permutation Problems and Well-Partially-Ordered Sets,” *Proceedings Fifth M.I.T. Conference on Advanced Research in VLSI*, published as *Advanced Research in VLSI* (J. Allen and F. T. Leighton, editors), The MIT Press, 1988, 315–327, with M. A. Langston.
3. “Fast Self-Reduction Algorithms for Combinatorial Problems of VLSI Design,” *Proceedings Third International Workshop on Parallel Computation and VLSI Theory*, Springer-Verlag, Lecture Notes in Computer Science vol. 319 (1988), 278–287, with M. A. Langston.
4. “On Search, Decision and the Efficiency of Polynomial-Time Algorithms,” *Proceedings ACM Symposium on the Theory of Computing (STOC)* (1989), 501–512, with M. A. Langston.
5. “An Analogue of the Myhill-Nerode Theorem and Its Use in Computing Finite-Basis Characterizations,” *Proceedings Thirtieth IEEE Symposium on the Foundations of Computer Science (FOCS)* (1989), 520–525, with M. A. Langston.
6. “On the Complexity of Fixed-Parameter Problems,” *Proceedings Thirtieth IEEE Symposium on the Foundations of Computer Science (FOCS)* (1989), 210–215, with K. Abrahamson, J. Ellis and M. Mata.
7. “Finite-Basis Theorems and a Computation-Integrated Approach to Obstruction Set Isolation,” *Proceedings of the First MIT Conference on Computers and Mathematics*, in *Computers and Mathematics* (E. Kaltofen and S.M. Watt, editors), Springer-Verlag (1989), 37–45, with N.G. Kinnersley and M.A. Langston.
8. “Computer Science in the Elementary Schools,” Mathematicians and Education Reform Workshop, Seattle, 1991. Proceedings published as: *Mathematicians and Education Reform 1990–1991*, N. Fisher, H. Keynes and P. Wagreich, eds., Conference Board of the Mathematical Sciences, *Issues in Mathematics Education* 3 (1993), 143–163.
9. “Finite Automata, Bounded Treewidth and Well-Quasiordering,” in: N. Robertson and P. Seymour (editors), *Graph Structure Theory: Proceedings of the Joint Summer Research Conference on Graph Minors, Seattle, June, 1991*, American Mathematical Society, *Contemporary Mathematics* 147 (1993), 539–564, with Karl Abrahamson.

10. "Algebraic Constructions of Efficient Broadcast Networks," in: H.F. Mattson, T. Mora and T.R.N. Rao (editors), *Proceedings of the Ninth International Symposium on Applied Algebra, Algebraic Algorithms and Error-Correcting Codes (AAECC'91)*, Springer-Verlag, Berlin, Lecture Notes in Computer Science, volume 539, pp. 152-158, with M. Dinneen and V. Faber.
11. "Two Strikes Against Perfect Phylogeny," in: W. Kuich (editor), *Proceedings of the 19th International Colloquium on Automata, Languages and Programming (ICALP'92)*, Springer-Verlag, Berlin, Lecture Notes in Computer Science, volume 623, pp. 273-283, with H. L. Bodlaender and T. J. Warnow.
12. "Parallel Self-Reducibility," *Proc. 4th International Conference on Computing and Information*, IEEE Computer Society Press (1992), 67-70, with K. Abrahamson and C. Wilson.
13. "Self-Witnessing Polynomial-Time Complexity and Certificates for Primal-ity," *Proceedings of the Seventh Annual IEEE Conference on Structure in Complexity Theory* (1992), 107-110, with N. Koblitz.
14. "Fixed-Parameter Intractability," *Proceedings of the Seventh Annual IEEE Conference on Structure in Complexity Theory* (1992), 36-49, with R. Downey.
15. "Kid Krypto," *Proceedings of Crypto '92*, Springer-Verlag, Lecture Notes in Computer Science vol. 740 (1993), 371-389, with N. Koblitz.
16. "Fixed-Parameter Tractability and Completeness III: Some Structural Aspects of the W -Hierarchy," *Proceedings of the 1992 Dagstuhl Workshop on Structural Complexity, Complexity Theory: Current Research*, ed. K. Ambos-Spies, S. Homer and U. Schöning, Cambridge University Press (1993), 191-226, with R.G. Downey.
17. "Parameterized Computational Feasibility," *Proceedings of the Second Cornell Workshop on Feasible Mathematics, Feasible Mathematics II*, P. Clote and J. Remmel (eds.), Birkhauser Boston (1995), 219-244, with R.G. Downey.
18. "Fixed-Parameter Intractability II," *Proceedings of the 10th Symposium on Theoretical Aspects of Computer Science (STACS'93)*, Springer-Verlag, Lecture Notes in Computer Science vol. 665 (1993), 374-385 with K. Abrahamson and R. Downey.
19. "Fixed-Parameter Complexity and Cryptography," *Proceedings of the Tenth International Symposium on Applied Algebra, Algebraic Algorithms and Error-Correcting Codes (AAECC'93)*, Springer-Verlag, Berlin, Lecture Notes in Computer Science vol. 673 (1993), 121-131, with N. Koblitz.
20. "Parameterized Learning Complexity," *Proceedings of the Sixth ACM Workshop on Computational Learning Theory (COLT'93)*, 51-57, with R.G. Downey and P.A. Evans.
21. "Advice Classes of Parameterized Tractability," *Proceedings of the Asian Logic Conference* (1993), with R. Downey, L. Cai and J. Chen.
22. "DNA Physical Mapping: Three Ways Difficult," in *Algorithms — ESA '93*, (Proceedings of the First European Symposium on Algorithms), Springer-Verlag, Berlin, Lecture Notes in Computer Science vol. 726 (1993), 157-168, with M.T. Hallett and H.T. Wareham.

23. "Combinatorial Cryptosystems Galore!" *Proceedings of the Second International Symposium on Finite Fields*, Las Vegas, Nevada, August, 1993, *Contemporary Mathematics* 168 (1994), 51–61, with N. Koblitz.
24. "On the Structure of Parameterized Problems in NP," *Proceedings of the 11th Symposium on Theoretical Aspects of Computer Science (STACS'94)*, Springer-Verlag, Lecture Notes in Computer Science vol. 775 (1994), 509–520, with L. Cai, J. Chen, and R. Downey.
25. "The Parameterized Complexity of Sequence Alignment and Consensus," *Proceedings of the Fifth Symposium on Combinatorial Pattern Matching (CPM)*, Springer-Verlag, Lecture Notes in Computer Science vol. 807 (1994), 15–30, with H. Bodlaender, R. Downey, and H. T. Wareham.
26. "Beyond NP-Completeness for Problems of Bounded Width: Hardness for the W Hierarchy," *Proceedings of the ACM Symposium on the Theory of Computing (STOC)* (1994), 449–458, with H. Bodlaender and M. Hallett.
27. "The Parameterized Complexity of Some Problems in Logic and Linguistics," *Proceedings Symposium on Logical Foundations of Computer Science (LFCS)*, Springer-Verlag, Lecture Notes in Computer Science vol. 813 (1994), 89–100, with R. Downey, B. Kapron, M. Hallett and H. T. Wareham.
28. "Parameterized Complexity Analysis in Computational Biology," *Proceedings of the IEEE Computer Society Workshop on Shape and Pattern Recognition in Computational Biology*, Seattle, June 1994, IBM TJ Watson Research Center Publication (1994), 99–116, with H. Bodlaender, R.G. Downey, M.T. Hallett and H.T. Wareham. To be published by Plenum Press.
29. "Obstructions to Within a Few Vertices or Edges of Acyclic," *Proceedings WADS'95*, Springer-Verlag, Lecture Notes in Computer Science vol. 955 (1995), 415–427, with K. Cattell and M.J. Dinneen.
30. "Let's Focus on the First Four," with N. Casey in: *Discrete Mathematics in the Schools: How Can We Have an Impact?* (D. Franzblau, F. Roberts and J. Rosenstein, eds.) DIMACS/AMS proceedings series, 1997.
31. "Finite-State Computability of Annotations of Strings and Trees," *Proceedings Seventh Symposium on Combinatorial Pattern Matching (CPM '96)*, Springer-Verlag, Lecture Notes in Computer Science vol. 1075 (1996), 384–391, with H. Bodlaender and P. Evans.
32. "The Heart of Puzzling: Mathematics and Computer Games," *Proceedings of the 1996 Computer Games Developers Conference*, Miller Freeman (1996), 109–120.
33. "The Parameterized Complexity of Relational Database Queries and An Improved Characterization of $W[1]$," in: *Combinatorics, Complexity and Logic*, Proceedings of DMTCS '96, (D. Bridges, C. Calude, J. Gibbons, S. Reeves, and I. Witten, Eds.) Springer-Verlag (1996), 194–213, with R. Downey and U. Taylor.
34. "Descriptive Complexity and the W Hierarchy," in: *Proof Complexity and Feasible Arithmetics* (P. Beame and S. Buss, Eds.) AMS-DIMACS Series in Discrete Mathematics and Theoretical Computer Science, American Mathematical Society (1997), 119–134, with R. Downey and K. Regan.

35. "Parameterized Complexity: A Framework for Systematically Confronting Computational Intractability," in: *Contemporary Trends in Discrete Mathematics*, (R. Graham, J. Kratochvíl, J. Nešetřil and F. Roberts, eds.), Proc. DIMACS-DIMATIA Workshop, Prague, 1997, *AMS-DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, vol. 49 (1999), 49-99, with R. Downey and U. Stege.
36. "Analogues and Duals of the MAST Problem for Sequences and Trees," *Proceedings of the Sixth European Symposium on Algorithms – ESA '98*, Springer-Verlag Lecture Notes in Computer Science, vol. 1461 (1998), 103–114, with M. Hallett, C. Korostensky and U. Stege.
37. "On the Multiple Gene Duplication Problem," *Proceedings Ninth International Symposium on Algorithms and Computation – ISAAC'98*, Springer-Verlag Lecture Notes in Computer Science, vol. 1533 (1998), 347–356, with M. Hallett and U. Stege.
38. "Explaining Cryptographic Systems to the General Public," *Proc. First IFIP World Conference on Information Security Education (WISE)*, L. Yngstgröm and S. Fischer-Hübner (eds.), Stockholm University Report Series 99-008 (1999), 221-233, with T. Bell, I. Witten and N. Koblitz.
39. "Parameterized Complexity After (Almost) 10 Years: Review and Open Questions," in: *Combinatorics, Computation and Logic, DMTCS'99 and CATS'99*, Australian Computer Science Communications 21, Springer-Verlag Singapore (1999), 1–33, with R.G. Downey.
40. "Coordinatized Kernels and Catalytic Reductions: An Improved FPT Algorithm for Max Leaf Spanning Tree and Other Problems," *Proc. FST-TCS 2000*, Springer-Verlag, *Lecture Notes in Computer Science* 1974 (2000), 240–251, with C. McCartin, F. Rosamond and U. Stege.
41. "Refined Search Tree Techniques for the Dominating Set Problem on Planar Graphs," *Proc. 26th International Symposium on Mathematical Foundations of Computer Science (MFCS 2001)*, Springer-Verlag, *Lecture Notes in Computer Science* 2136 (2001), 111-122, with J. Alber, H. Fan, H. Fernau, R. Niedermeier, F. Rosamond and U. Stege.
42. "A Fixed-Parameter Approach to Two-Layer Planarization," *Proc. 9th International Symposium on Graph Drawing (GD 2001)*, Springer-Verlag, *Lecture Notes in Computer Science* 2265 (2001), 1–15, with V. Dujmovic, M. Hallett, M. Kitching, G. Liotta, C. McCartin, N. Nishimura, P. Ragde, F. Rosamond, M. Suderman, S. Whitesides and D. Wood.
43. "On the Parameterized Complexity of Layered Graph Drawing," *Proc. 9th Annual European Symposium on Algorithms (ESA 2001)*, Springer-Verlag, *Lecture Notes in Computer Science* 2161 (2001), 488–499, with V. Dujmovic, M. Hallett, M. Kitching, G. Liotta, C. McCartin, N. Nishimura, P. Ragde, F. Rosamond, M. Suderman, S. Whitesides and D. Wood.
44. "Parameterized Complexity: New Developments and Research Frontiers," *Proc. New Zealand Mathematical Sciences Research Institute Summer Workshop*, Kaikoura, 2000, *Aspects of Complexity*, R. Downey and D. Hirschfeldt (eds.), de Gruyter (2001), 51–72 (notes on featured short course).

45. “Some New Developments in Parameterized Complexity,” *Proc. 12th Australasian Workshop on Combinatorial Algorithms*, ed. Edy Tri Baskoro (2001), 43–44.
46. “Parameterized Complexity: Main Ideas, Connections to Heuristics and Research Frontiers,” *Proc. ISAAC 2001*, Springer-Verlag, *Lecture Notes in Computer Science* 2223 (2001), 291–307.
47. “Parameterized Complexity: The Main Ideas and Connections to Practical Computing,” *Proc. CATS 2002, Computing: The Australasian Theory Symposium*, James Harland (ed.), Elsevier, *Electronic Notes in Computer Science* 61 (2002), 1–17.
48. On the “Parameterized Intractability of CLOSEST SUBSTRING and Related Problems,” *Proc. STACS 2002*, Springer-Verlag, *Lecture Notes in Computer Science* 2285 (2002), 262–273, with J. Gramm and R. Niedermeier.
49. “Parameterized Complexity: The Main Ideas and Connections to Practical Computing.” In *Experimental Algorithmics*, Springer-Verlag, *Lecture Notes in Computer Science* 2547 (2002), 51–77.
50. “Efficient Data Reduction for DOMINATING SET: A Linear Problem Kernel for the Planar Case,” *Proc. SWAT 2002*, Springer-Verlag, *Lecture Notes in Computer Science* 2368 (2002), 150–159, with J. Alber and R. Niedermeier.
51. “The Dominating Set Problem is Fixed Parameter Tractable on Graphs of Bounded Genus,” *Proc. SWAT 2002*, Springer-Verlag, *Lecture Notes in Computer Science* 2368 (2002), 180–189, with J. Ellis and H. Fan.
52. “Blow-Ups, Win/Win’s and Crown Rules: Some New Directions in FPT,” *Proceedings WG 2003*, Springer-Verlag, *Lecture Notes in Computer Science* 2880 (2003), 1–12.
53. “An FPT Algorithm for Set Splitting,” *Proceedings WG 2003*, Springer-Verlag, *Lecture Notes in Computer Science* 2880 (2003), 180–191, with F. Dehne and F. Rosamond.
54. “New Directions and New Challenges in Algorithm Design and Complexity, Parameterized,” *Proceedings WADS 2003*, Springer-Verlag, *Lecture Notes in Computer Science* 2748 (2003), 505–520.
55. “Starting with Nondeterminism: the Systematic Derivation of Linear-Time Graph Layout Algorithms,” *Proceedings MFCS 2003*, Springer-Verlag, *Lecture Notes in Computer Science* 2747 (2003), 239–248, with H. Bodlaender and D. Thilikos.
56. “Kernelization Algorithms for the Vertex Cover Problem: Theory and Experiments,” *Proceedings ALLENEX/ANALC 2004*, Springer-Verlag, *Lecture Notes in Computer Science* (2004), 62–69, with F. Abu-Khzam, R. Collins, M. Langston and W. H. Suters.
57. “Tight Lower Bounds for Certain Parameterized NP-hard Problems,” *Proceedings of the IEEE Conference on Computational Complexity* (2004), 150–160, with J. Chen, B. Chor, X. Huang, D. Juedes, I. Kanj and G. Xia.

58. “Greedy Localization, Iterative Compression and Modeled Crown Reductions: New FPT Techniques, an Improved FPT Algorithm for Set Splitting, and a Novel $2k$ Kernelization for Vertex Cover,” *Proceedings of the First International Workshop on Parameterized and Exact Computation (IWPEC 2004)*, Springer-Verlag, *Lecture Notes in Computer Science* 3162 (2004), 271–282, with F. Dehne, F. Rosamond and P. Shaw.
59. “A Survey of FPT Algorithm Design Techniques with an Emphasis on Recent Advances and Connections to Practical Computing,” *Proceedings ESA 2004*, Springer-Verlag, *Lecture Notes in Computer Science* 3221 (2004), 1–2.
60. “On Finding Short Resolution Refutations and Small Unsatisfiable Subsets,” *Proceedings of the First International Workshop on Parameterized and Exact Computation (IWPEC 2004)*, Springer-Verlag, *Lecture Notes in Computer Science* 3162 (2004), 223–234, with S. Szeider and G. Wrightson.
61. “Faster Fixed-Parameter Tractable Algorithms for Matching and Packing Problems,” *Proceedings ESA 2004*, Springer-Verlag, *Lecture Notes in Computer Science* 3221 (2004), 311–322, with C. Knauer, N. Nishimura, P. Ragde, F. Rosamond, U. Stege, D. Thilikos and S. Whitesides.
62. “Finding k Disjoint Triangles in an Arbitrary Graph,” *Proceedings WG 2004*, Springer-Verlag, *Lecture Notes in Computer Science* 3353 (2004), 235–244, with P. Heggernes, F. Rosamond, C. Sloper and J.-A. Telle.
63. “Linear Kernels in Linear Time, or How to Save k Colors in $O(n^2)$ Steps,” *Proceedings WG 2004*, Springer-Verlag, *Lecture Notes in Computer Science* 3353 (2004), 257–269, with B. Chor and D. Juedes.
64. “An $O(2^{O(k)}n^3)$ FPT Algorithm for the Undirected Feedback Vertex Set Problem,” *Proceedings COCOON 2005*, Springer-Verlag, *Lecture Notes in Computer Science* 3595 (2005), 859–869, with F. Dehne, M. Langston, F. Rosamond and K. Stevens.
65. “Fixed-Parameter Tractability is Polynomial-Time Extremal Structure Theory I: The Case of Max Leaf,” *Proceedings of ACiD 2005: Algorithms and Complexity in Durham*, Kings College London Publications, *Texts in Algorithmics* 4 (2005), 1–41, with V. Estivill-Castro, M. Langston and F. Rosamond.
66. “Nonblocker: Parameterized Algorithmics for Minimum Dominating Set,” *Proceedings SOFSEM 2006: 32nd Conference on Current Trends in Theory and Practice of Computer Science*, Springer-Verlag, *Lecture Notes in Computer Science* 3831 (2006), 237–245, with F. Dehne, H. Fernau, E. Prieto and F. Rosamond.
67. “Cliquewidth Minimization is NP-hard,” *Proceedings of the ACM Symposium on Theory of Computing* (2006), 354–362, with F. Rosamond, U. Rotics and S. Szeider.
68. “The Undirected Feedback Vertex Set Problem has Polynomial Kernel Size,” *Proceedings IWPEC 2006*, Springer-Verlag, *Lecture Notes in Computer Science* 4169 (2006), 192–202, with K. Burrage, V. Estivill-Castro, M. Langston, S. Mac and F. Rosamond.

69. "Parameterized Approximation Problems," *Proceedings IWPEC 2006*, Springer-Verlag, *Lecture Notes in Computer Science* 4169 (2006), 121–129, with R. Downey and C. McCartin.
70. "The Lost Continent of Polynomial Time," *Proceedings IWPEC 2006*, Springer-Verlag, *Lecture Notes in Computer Science* 4169 (2006), 276–277.
71. "Kernelization for Convex Recoloring of Trees," *Proc. ACiD 2006*, King's College Publications, *Texts in Algorithmics* 7 (2006), 23–36, with H. Bodlaender, M. Langston, M. Ragan and F. Rosamond.
72. "On the Complexity of Lobbying in Multiple Referenda," *Proc. First International Workshop on Computational Social Choice*, pp. 87–96, (Amsterdam, Dec. 2006) with R. Christian, F. Rosamond and A. Slinko.
73. "Why Is P Not Equal to NP?" *Computation and Logic in the Real World*, Third Conference on Computability in Europe, CiE 2007, Siena, June 2007, Local Proceedings (Technical Report 487, Dipartimento di Scienze Matematiche ed Informatiche, Universita Degli Studi Di Siena), pp. 151–160, with F. Rosamond.
74. "The Complexity Ecology of Parameters: An Illustration Using Bounded Max Leaf Number," *Proceedings of CiE 2007*, Springer-Verlag, *Lecture Notes in Computer Science* 4497 (2007), 268–277, with F. Rosamond.
75. "Quadratic Kernelization for Convex Recoloring of Trees," *Proceedings of COCOON 2007*, Springer-Verlag, *Lecture Notes in Computer Science* 4598 (2007), 86–96, with H. Bodlaender, M. Langston, M. Ragan, F. Rosamond and M. Weyer.
76. "Connected Coloring Completion for General Graphs: Algorithms and Complexity," *Proceedings of COCOON 2007*, Springer-Verlag, *Lecture Notes in Computer Science* 4598 (2007), 75–85, with B. Chor, M. Ragan, I. Razgon, F. Rosamond and S. Snir.
77. "Sharp Tractability Borderlines for Finding Connected Motifs in Vertex-Colored Graphs," *Proceedings of ICALP 2007*, Springer-Verlag, *Lecture Notes in Computer Science* 4596 (2007), 340–351, with G. Fertin, D. Hermelin and S. Vialette.
78. "Efficient Parameterized Preprocessing for Cluster Editing," *Proceedings of FCT 2007*, Springer-Verlag, *Lecture Notes in Computer Science* 4639 (2007), 312–321, with M. Langston, F. Rosamond and P. Shaw.
79. "On the Complexity of Some Colorful Problems Parameterized by Treewidth," *Proceedings of COCOA 2007*, Springer-Verlag, *Lecture Notes in Computer Science* 4616 (2007), 366–377, with F. Fomin, D. Lokshtanov, F. Rosamond, S. Saurabh, S. Szeider and C. Thomassen.
80. "Parameterized Complexity via Combinatorial Circuits," *Proc. 3rd ACiD*, 2007, King's College Publications, London, *Texts in Algorithmics* 9 (2007), 55–67.
81. "Fixed-Parameter Algorithms for Kemeny Scores," *Proceedings of AAIM 2008*, Springer-Verlag, *Lecture Notes in Computer Science* 5034 (2008), 60–71, with N. Betzler, J. Guo, R. Niedermeier and F. Rosamond. Invited for submission to a special issue of *Theoretical Computer Science*.

82. "A Purely Democratic Characterization of $W[1]$," *Proceedings of IWPEC 2008*, Springer-Verlag, *Lecture Notes in Computer Science* 5018 (2008), 103–114, with D. Hermelin, M. Müller and F. Rosamond.
83. "Facilities Location Problems: A Parameterized View," *Proceedings of AAIM 2008*, Springer-Verlag, *Lecture Notes in Computer Science* 5034 (2008), 188–199, with H. Fernau.
84. "Parameterized Algorithms and Hardness Results for Some Graph Motif Problems," CPM 2008, *Lecture Notes in Computer Science* 5029 (2008), 31–43, with N. Betzler, C. Komusiewicz and R. Niedermeier.
85. "On Problems Without Polynomial Kernels," ICALP 2008, *Lecture Notes in Computer Science* 5125 (2008), 563–574, with H. Bodlaender, R. Downey and D. Hermelin.
86. "Clustering with Partial Information," MFCS 2008, *Lecture Notes in Computer Science* 5162 (2008), 144–155, with H.L. Bodlaender, P. Heggernes, F. Mancini, C. Papadopoulos and F. Rosamond.
87. "Computing Kemeny Rankings, Parameterized by the Average K-T Distance," COMSOC 2008, with N. Betzler, J. Guo, R. Niedermeier and F. Rosamond.
88. "Graph Layout Problems Parameterized by Vertex Cover," *International Symposium on Automata, Algorithms and Computation*, ISAAC 2008, *Lecture Notes in Computer Science* 5369 (2008), 294–305, with D. Lokshtanov, N. Misra, F. Rosamond and S. Saurabh.
89. "Leaf Powers and Their Properties: Using the Trees," ISAAC 2008, *Lecture Notes in Computer Science* 5369 (2008), 402–413. with D. Meister, F. Rosamond, R. Sritharan and J.A. Telle.
90. "Parameterized Complexity of Stabbing Rectangles and Squares in the Plane," Third Workshop on Algorithms and Computation, WALCOM 2009, *Lecture Notes in Computer Science* 5431 (2009), 298–309, with M. Dom and F. Rosamond.
91. "A Generalization of Nemhauser and Trotter's Local Optimization Algorithm," *Proceedings STACS 2009*, 409–420, with J. Guo, H. Moser and R. Niedermeier.
92. "How Similarity Helps to Efficiently Compute Kemeny Rankings," *Proceedings 8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2009)*, 657–664, with N. Betzler, J. Guo, R. Niedermeier and F. Rosamond.
93. "Haplotype Inference Constrained by Plausible Haplotype Data," *Proceedings CPM 2009*, Springer-Verlag, *Lecture Notes in Computer Science* 5577 (2009), 339–352, with T. Hartman, D. Hermelin, G. Landau, L. Leventhal and F. Rosamond.
94. "Local Search: Is Brute Force Avoidable?" *Proceedings International Joint Conference on Artificial Intelligence*, IJCAI (2009), 486–491, with F. Fomin, D. Lokshtanov, F. Rosamond, S. Saurabh and Y. Villanger.
95. "Graph-Based Data Clustering with Overlaps," COCOON 2009, Springer-Verlag, *Lecture Notes in Computer Science* 5609 (2009), 516–526, with J. Guo, C. Komusiewicz, R. Niedermeier and J. Uhlmann.

96. “Distortion Is Fixed-Parameter Tractable,” ICALP 2009, Springer-Verlag, *Lecture Notes in Computer Science* 5555 (2009), 463–474, with F. Fomin, D. Lokshтанov, E. Losievskaja, F. Rosamond and S. Saurabh.
97. “The Parameterized Complexity of Some Minimum Label Problems,” Proceedings WG 2009, Springer-Verlag, *Lecture Notes in Computer Science* 5911 (2009), 88–99, with J. Guo and I. Kanj.
98. “Well-Quasi-Ordering Bounded Treewidth Graphs,” Proceedings IWPEC 2009, Springer-Verlag, *Lecture Notes in Computer Science* 5917 (2009), 149–160, with D. Hermelin and F. Rosamond.
99. “A Complexity Dichotomy for Finding Disjoint Solutions of Vertex Deletion Problems,” Proceedings MFCS 2009, Springer-Verlag, *Lecture Notes in Computer Science* 5734 (2009), 319–330, with J. Guo, H. Moser and R. Niedermeier.
100. “What Makes Equitable Connected Partition Easy?” Proceedings IWPEC 2009, Springer-Verlag, *Lecture Notes in Computer Science* 5917 (2009), with R. Enciso, J. Guo, I. Kanj, F. Rosamond and A. Suchy.
101. “Towards Fully Multivariate Algorithmics: Some New Results and Directions in Parameter Ecology,” Proceedings IWOCА 2009, Springer-Verlag, *Lecture Notes in Computer Science* 5874 (2009), 2–10.
102. “Fixed-Parameter Tractability, Relative Kernelization and the Effectivization of Structural Connections,” CiE 2009, with J. Hromkovic, F. Rosamond and M. Steinova.
103. “Milling a Graph with Turn Costs: A Parameterized Complexity Perspective,” Proceedings WG 2010, Springer-Verlag, *Lecture Notes in Computer Science* 6410 (2010), 123–134, with P. Giannopoulos, C. Knauer, C. Paul, F. Rosamond, S. Whitesides and N. Yu.
104. “A Linear Kernel for Co-Path/Cycle Packing,” Proceedings of AAIM 2010, Springer-Verlag, *Lecture Notes in Computer Science* 6124 (2010), 90–102, with Z-Z. Chen, B. Fu, H. Jiang, Y. Liu, L. Wang and B. Zhu.
105. “Parameterized Control Complexity in Bucklin Voting and in Fallback Voting,” Proceedings COMSOC 2010, with G. Erdelyi.
106. “Parameterized Hardness of Dodgson Score,” Proceedings FST-TCS 2010, *Leibniz International Proceedings in Informatics* (2010), 459–468, with B. Jansen, D. Lokshтанov and S. Saurabh.
107. “Parameterizing by the Number of Numbers,” Proceedings IPEC 2010, Springer-Verlag, *Lecture Notes in Computer Science* 6478 (2010), 123–134, with S. Gaspers and F. Rosamond.
108. “Recent Developments in the Theory of Pre-Processing,” Proceedings FAW/AAIM 2011, Springer-Verlag, *Lecture Notes in Computer Science* 6681 (2011), 4–5.
109. “Constraint Satisfaction Problems: Convexity Makes AllDifferent Constraints Tractable,” Proceedings IJCAI 2011: 522–527, with T. Friedrich, D. Hermelin, N. Narodytska and F. Rosamond.
110. “Parameterized Complexity of the Firefighter Problem,” Proceedings ISAAC 2011, Springer-Verlag, *Lecture Notes in Computer Science* 7074 (2011), 643–652, with C. Bazgan, and M. Chopin.

111. “Simultaneously Satisfying Linear Equations Over $F[2]$: MaxLin2 and Maxr-Lin2 Parameterized Above Average,” *Proceedings FST-TCS 2011* (Schloss Dagstuhl – Leibniz Centrum fuer Informatik): 229–240, with R. Crowston, G. Gutin, M. Jones, F. Rosamond, S. Thomasse and A. Yeó.
112. “Train Marshalling is Fixed Parameter Tractable,” accepted to AAC 2012, with no proceedings, and to FUN 2012, with proceedings, with L. Brueggeman, R. Fleischer, M. Lackner, C. Komusiewicz, Y. Koutis, A. Pfandler and F. Rosamond.
113. “The Parameterized Complexity of Abduction,” to appear in Proceedings AAAI 2012, with A. Pfandler, F. Rosamond and S. Ruemmele.

Recent Invited Conference Presentations

- “Parameterized Complexity,” New Zealand Mathematics Society Annual Summer Workshop, Featured Short Course, January 2000.
- “Parameterized Complexity,” Twelfth International Symposium on Algorithms and Computation (ISAAC 2001), December 2001, Christchurch, New Zealand (Invited Plenary Lecture).
- “Parameterized Complexity and Its Applications,” Invited Plenary Lecture at CATS 2002, Melbourne.
- Invited Plenary Lecture at WADS 2003.
- Invited Plenary Lecture at WG 2003.
- Invited Plenary Lecture at ESA 2004.
- Invited Plenary Lecture at the *Algorithms and Complexity in Durham* Workshop, July, 2005.
- Invited Special Lecture at the Third Dagstuhl Workshop on Parameterized Complexity, July, 2005.
- Invited Plenary Lecture at IWPEC 2006, Zurich, September 2006.
- Special Opening Lecture at the Fourth Dagstuhl Workshop on Parameterized Complexity, July, 2007.
- Invited Plenary Lecture at ICYCS 2008, Yunan, China, November, 2008.
- Invited Special Lecture at the Dagstuhl Workshop on Communication of Computer Science (“The Brainware Crisis”), March, 2009.
- Invited Lecturer, AGAPE Summer School on Parameterized and Exact Algorithms, Corsica, May, 2009.
- Featured International Research Colloquium Speaker, Chinese University of Hong Kong, April, 2009.
- Invited Plenary Lecture at IWOCA 2009, Czech Republic, July, 2009.
- Invited Plenary Lecture at ACCMCC 2010, December, Canberra, Australia.
- Featured Invited Lecture, DIMAP, Warwick University, UK, April, 2011.
- Invited Plenary Speaker, FAW/AAIM 2011, Jinhua, China, May, 2011.
- Keynote Address, WORKER 2011, Vienna, Austria, September, 2011.

Professional Service and Honors

- Associate Editor for the *Journal of Computer and Systems Sciences*.
- Associate Editor for the *ACM Transactions on Algorithms*.
- Guest Editor for a double special issue of *The Computer Journal* in 2008 (Numbers 1,3).
- Guest Editor for a special issue of *Discrete Optimization* 8 (2011).
- Member of the Steering Committee for the conference series *International Workshop on Parameterized and Exact Computation*, 2002–2012.
- Recipient of an Erskine Fellowship with the Department of Computer Science at the University of Canterbury, 1996.
- Recipient of a Fellowship to the Institute of Advanced Study, Durham University, January–March, 2007.
- Recipient of a Humboldt Research Award, 2007.
- Recipient of a Australian Professorial Fellowship, 2010–2014.
- Appointment with the title “Visiting Professor in Computer Science,” to the Royal Holloway, University of London, 2009–2011.

Conference Organization

- First Idaho ONR Workshop on Software Research, June, 1989, Conference Chair.
- Second Idaho ONR Workshop on Software Research, June, 1990, Conference Chair.
- STOC '92 Conference Chair.
- Co-organizer, Dagstuhl Workshop on Parameterized Complexity, August 2001.
- Co-organizer, Workshop on Structural Aspects of Parameterized Complexity, in conjunction with FST-TCS 2002, Kanpur, India, December 2002.
- Co-organizer, Dagstuhl Workshop on Parameterized Algorithms, July 2003.
- Co-chair, First International Workshop on Parameterized and Exact Computation, Bergen, 2004.
- Co-organizer, Dagstuhl Workshop on Parameterized Complexity and Kernelization, June, 2012.

Program Committees

DMTCS 2002, FST-TCS 2002, COCOON 2003, WADS 2003, CATS 2003, ACSW 2003, CATS 2004, ACSW 2004, WG 2004, IWPEC 2004 (program co-chair), MFCS 2005, ACSW 2005, IWPEC 2006, WG 2008, COCOA 2008, FAW 2008, ICYCS 2008 (program co-chair), ALENEX 2009, IWPEC 2009, TAMC 2009, FAW 2009, COMSOC 2010, IWOCA 2010, LATA 2010, IPEC 2010, TAMC 2012, APEX 2012, MFCS 2012.

Graduate Student Supervision

- Mark Hoover, Ph.D., 1989.
- Yasu Koda, Ph.D., 1991.
- Xiuyan Liu, M.S., 1994.
- Michael Dinneen, Ph.D., 1996.
- Michael Hallett, Ph.D., 1996.
- Todd Wareham, Ph.D., 1997.
- Patricia Evans, Ph.D., 1999.
- Elena Prieto-Rodriguez, Ph.D., 2005.

Postdoctoral Student Supervision

- Ulrike Stege

Miscellaneous

- Google scholar citations for *Michael Fellows*: 11027
- Google scholar H-index for *Michael Fellows*: 50
- Erdős number of Mike Fellows: 2 (via N. Alon, L. Clark, R. Entinger, V. Faber, F. Harary, S. Hedetniemi, P. Hell, D. Kleitman, C. Thomassen)

Table of Contents

Part I: Memories

Fixed-Parameter Tractability, A Prehistory	3
<i>Michael A. Langston</i>	
The Birth and Early Years of Parameterized Complexity	17
<i>Rod Downey</i>	
Crypto Galore!	39
<i>Neal Koblitz</i>	
Flyby: Life Before, During, and After Graduate Studies with Mike Fellows	51
<i>Todd Wareham</i>	
The Impact of Parameterized Complexity to Interdisciplinary Problem Solving	56
<i>Ulrike Stege</i>	
Vertex Cover, Dominating Set and My Encounters with Parameterized Complexity and Mike Fellows	69
<i>Venkatesh Raman</i>	
Mike Fellows: Weaving the Web of Mathematics and Adventure	74
<i>Jan Arne Telle</i>	
Passion Plays: Melodramas about Mathematics	80
<i>Frances Rosamond</i>	

Part II: Surveys

A Basic Parameterized Complexity Primer	91
<i>Rod Downey</i>	
Kernelization – Preprocessing with a Guarantee	129
<i>Daniel Lokshтанov, Neeldhara Misra, and Saket Saurabh</i>	
Parameterized Complexity and Subexponential-Time Computability	162
<i>Jianer Chen and Iyad A. Kanj</i>	
Fixed-Parameter Tractability of Treewidth and Pathwidth	196
<i>Hans L. Bodlaender</i>	
Graph Minors and Parameterized Algorithm Design	228
<i>Dimitrios M. Thilikos</i>	

Constraint Satisfaction Problems Parameterized above or below Tight Bounds: A Survey	257
<i>Gregory Gutin and Anders Yeo</i>	
Backdoors to Satisfaction	287
<i>Serge Gaspers and Stefan Szeider</i>	
Studies in Computational Aspects of Voting: A Parameterized Complexity Perspective	318
<i>Nadja Betzler, Robert Brederbeck, Jiehua Chen, and Rolf Niedermeier</i>	
A Parameterized Halting Problem	364
<i>Yijia Chen and Jörg Flum</i>	
Computer Science Unplugged and Related Projects in Math and Computer Science Popularization	398
<i>Tim Bell, Frances Rosamond, and Nancy Casey</i>	
FPT Suspects and Tough Customers: Open Problems of Downey and Fellows	457
<i>Fedor V. Fomin and Dániel Marx</i>	
What's Next? Future Directions in Parameterized Complexity	469
<i>Dániel Marx</i>	
Author Index	497

Part I
Memories

Fixed-Parameter Tractability, A Prehistory^{*,**}

*A Festschrift Contribution Devoted to Michael R. Fellows
on the Occasion of his 60th Birthday*

Michael A. Langston

Department of Electrical Engineering and Computer Science

University of Tennessee
Knoxville, TN 37996-2250
USA

langston@eecs.utk.edu

1 Overview

Many of the foundational parameterized tenets discussed in this festschrift actually predate by over a decade the first systematic treatments of fixed-parameter tractability. In this frank, firsthand account I will, to the best of my recollection, describe some of the earliest research avenues Mike Fellows and I pursued that would turn out later to be highly relevant to parameterized complexity. Although we did not know it at the time, these were the origins and formative years of this burgeoning new field. Readers unfamiliar with the history of fixed-parameter tractability may be surprised to learn that its initial motivations arose from, of all things, automation and optimization for integrated circuit design.

2 A Fortuitous Collaboration

I first met Mike Fellows sometime in the spring of 1985, the year he completed his PhD in Computer Science at the University of California, San Diego. I was then chairing the faculty search committee at Washington State University, where we were fortunate enough to interview and hire him. Mike and I hit it off immediately. We had similar research interests in graph theory, combinatorics and optimization. We both had families with small children. And we both had even served in paratrooper assignments with the US military (Mike as an enlisted man in the Air Force, I as an officer in the Army). As luck would have it, I happened to be working at the time on a spectrum of combinatorial problems motivated by

* Prehistory (from the Latin, with *præ* meaning before, and *historia* meaning story) is often defined as the period before a story is recorded. And that is what this tale is all about. It is an account of the genesis of fixed-parameter tractability, before the field had its terminology or even its name.

** This narrative account was made possible in part by the National Science Foundation under grants MIP-8703879 and MIP-8919312, and by the Office of Naval Research under contract N00014-90-J-1855.

circuit layout problems in very large scale integration (VLSI) design. Meanwhile, Mike had been reading the recent work of Neil Robertson and Paul Seymour on Wagner’s Conjecture and what is now known as the Graph Minor Theorem. It was a perfect confluence of technologies and ideas. As soon as Mike arrived on campus, we put our heads together, compared notes, and decided to look around the VLSI domain to see if we could find any interesting lower ideals in the minor order. There was nothing particularly out of the ordinary with so inauspicious a start. I think we viewed it as a fairly routine academic exercise. We were merely searching for new research horizons. We could scarcely have foreseen where this journey would eventually take us. Thus it all began.

3 Research Atmosphere

I think the algorithmic landscape at that time was relatively complacent. Most problems of interest had already been found either to reside in \mathcal{P} or to be \mathcal{NP} -complete. Thus, natural problems were largely viewed under the classic Jack Edmonds style dichotomy as being good or bad, easy or hard, with not much of a middle ground. Most of our colleagues in the theoretical computer science community seemed pretty satisfied with this simple picture. Exhaustive exact techniques and heuristic approximation algorithms remained the stalwart analytical weapons of choice. I confess that I had in fact worked on the worst-case analysis of scheduling heuristics myself as part of my PhD dissertation a few years earlier. Nevertheless, new possibilities beckoned. Mike and I often wondered out loud why all \mathcal{NP} -complete problems were generally being tarred with the same brush. It made little sense to us. Sure, there were notions like strong \mathcal{NP} -completeness [8]. But where was any systematic sort of focus on the parameter effect? Even well-known \mathcal{NP} -complete problems like independent set and graph coloring were manifestly different, starting with parameters set as low as 3. Sorting out differences in problem difficulty based on parameter specifications was a theme that would keep working its way into, and eventually take over, much of our research program.

4 Changes Brewing

Meanwhile, things were quickly evolving in the worlds of extremal graph theory and well-quasi-ordered sets. Mike and I were I think a little awe struck with the possibilities. With mere checks on various forms of graph containment, one could now quickly show that many seemingly intractable problems, some not previously even known to be decidable, were actually in \mathcal{P} . Wow!

But let me not get ahead of myself. In what follows I shall consider only finite, undirected graphs. H is a *minor* of G if a graph isomorphic to H can be obtained from a subgraph of G by contracting edges. A family F of graphs is said to be *closed* in the minor order if every minor of a graph in F must also reside in F . A graph is an *obstruction* to F if it lies in F ’s complement and all its proper minors lie in F . Surely the most classic and widely-known example

is planarity, where F denotes the set of planar graphs, which is minor closed, and where the obstructions have long been known to be $K_{3,3}$ and K_5 . In the interest of historical accuracy, it should probably be pointed out that planarity was first studied in the *topological* order [12]. (H is topologically contained in G if a graph isomorphic to H can be obtained from a subgraph of G by removing subdivisions.) The obstruction sets for both orders are the same.

The most useful tool to us back then was this:

Theorem 1. [15] *Any family of finite graphs that is closed in the minor order and that excludes a planar graph can be recognized in polynomial time.*

At this point a simple example of a family amenable to this powerful theorem may be helpful. So let's consider vertex cover, which is probably the most widely studied problem in all of parameterized complexity. Let k denote any fixed positive integer. I will leave it as an exercise for the reader to check that (1) if G has a vertex cover of size k , then so does any of its minors and (2) there are planar graphs whose vertex cover size exceeds k . Thus, by Theorem 1, there is a polynomial-time algorithm to decide whether an arbitrary graph has a vertex cover of size k . It is important to note that there will be a different algorithm for each fixed k . Of course vertex cover is solvable by brute force in $O(n^k)$ time anyway, but soon Theorem 1 was superseded by even better tools [16,17,18], limiting the degree of the polynomial to 3. Now things are beginning to sound a lot like what would eventually become fixed-parameter tractability. Incidentally, these improvements also eliminated the need for planar exclusion, and added the *immersion* order to the mix (H is immersed in G if a graph isomorphic to H can be obtained from a subgraph of G by lifting pairs of edges). But I digress; let me return to the main story.

To implement the algorithm ensured by Theorem 1, one needs only to take as input an arbitrary graph and test whether it contains any of the family's obstructions. If it does, then the algorithm is to answer "no." Otherwise, it is to answer "yes." The run time guarantee comes from the facts that containment for each obstruction can be tested in polynomial time, and that only a finite battery of such tests is ever necessary³.

Of course there are substantial prices to be paid in applying such stunningly abstract and powerful strategies. The specific algorithms provided by these emergent tools impose ginormous and utterly impractical constants of proportionality. Worse yet, they are inherently non-constructive in that each requires complete knowledge of the relevant obstruction set. (I will have more to say on these issues later.) Despite such grave practical drawbacks, however, these dramatic developments helped reinforce our misgivings about any homogeneity among \mathcal{NP} -complete problems. Indeed, it seemed more and more plausible to us that

³ An order is a well-quasi-order if any infinite collection of objects must have a pair of comparable elements under that order. By what is now called the Graph Minor Theorem, we know that finite, simple graphs are well-quasi-ordered under minors. Thus, because obstructions are incomparable (they form an antichain), their number must be finite.

not all \mathcal{NP} -complete problems were created equal, and that many might even be sliced into easier sub-problems with the use appropriate parameters. Mike and I frequently discussed the need for some sort of non-uniform notion of problem complexity, usually by keeping a problem constant but applying a distinct algorithm for each parameter value. Of course this is precisely what we do today with fixed-parameter algorithms, but none of that was formalized back then.

5 Shoulders of Giants

In our many interactions with Neil Robertson and Paul Seymour, it became increasingly clear that they had no more than a marginal interest in practicalities, applications and implementations. And why should they? They were hot on the trail of exceedingly profound advances in graph theory. Yet serious practical applications were some of the very topics that Mike and I felt were of central relevance to computation. So off we went, with Neil's and Paul's blessing. We had a clear field in which to plow our furrows. I think it was pretty natural for us to feel rather inadequate when stacked up against their deep and beautiful work (as well as their joint efforts with Robin Thomas and others). In this Mike was always self-effacing and humble, ever quick to point out that by performing algorithm mining on well-quasi-order theory we were standing on the shoulders of giants. One might even argue that Mike held out this attitude with a just bit too much zeal, as he often wound up unfairly downplaying or even dismissing a great deal of his own original and creative work in the process.

6 Armchair Polynomial Time

It wasn't long before Mike and I started being invited around the country to speak on our work. We took the Mike and Mike show on the road with alacrity. At about this time I went on sabbatical leave at the University of Illinois, but I wound up away on travel at least as much as I was in Champaign. Non-constructive algorithm design tools were a new thing, and many wanted to find out how they worked. On one jaunt, Mike was invited to give a talk at Princeton, where he spoke to a computer science audience. There he humorously christened our earliest effort "armchair polynomial time." As he rightly pointed out, it was sometimes embarrassingly easy to apply Theorem 1 and its extensions. To reinforce the point, he ended the presentation of each proof with a tiny glyph of an armchair. I liked it!

As previously mentioned, fixed k vertex cover is a handy example of an exceedingly straightforward application of Theorem 1. With much subsequent work, however, vertex cover has now migrated from $O(n^k)$ to $O(n^3)$ to $O(n \log n)$ and of course now all the way down to $O(n)$. See, for example, [1]. Many other problems have followed this trajectory. Probably a better example, therefore, and one to showcase the truly astonishing power of this general approach, is knotlessness [5]. Here we are asked whether a graph can be embedded in three dimensional space so that none of its cycles are knotted. This sounds quite difficult. Given

an arbitrary graph, no method is known just to bound the number of its embeddings that must be tested. Thus, without recourse to Theorem 1, knotlessness is in no obvious way even known to be decidable. With the use of minor closure, however, it is not at all difficult to show:

Theorem 2. [5] *Knotlessness can be decided in polynomial time.*

Notorious problems like knotlessness and others of its ilk (e.g., linklessness [5]) are further distinguished from the rest of the problems I will discuss by the fact that they have no obvious associated parameter(s). Thus, while a problem like vertex cover has a distinct finite obstruction set, and hence a distinct algorithm, for every fixed cover size, knotlessness has but a single forbidden set. (Of course one could probably parameterize by, say, the number of knots or the dimensionality of the embedding space, but as far as I know no one has looked into that.)

7 Circuit Layout Applications

While Mike and I found applications across many domains, we noticed early on that the field of circuit design abounds with combinatorial problems amenable to this general approach. One of the first problems we studied, and the only one I will discuss in any detail here, is gate matrix layout. This style was introduced in [14] for CMOS circuits. Solving the problem at its heart was, and is, a central step in circuit synthesis. It was known at the time that gate matrix layout was equivalent to various problems encountered in multiple PLA folding and the use of Weinberger and one-dimensional logic arrays [4]. Gate matrix layout was later shown also to be equivalent to pathwidth [6], vertex separation number [10] and several other graph metrics.

The problem can be stated as follows. We are given a Boolean matrix M and a positive integer k . We are asked whether the columns of M can be permuted so that, if in each row every 0 lying between the leftmost and rightmost 1 is changed to *, no column contains more than k 1's and *'s. In this formulation, rows denote electrical circuits, columns denote gates, and a * represents the fact that all gates within a circuit must be physically connected. Circuits are not permitted to overlap within a track. Minimizing the maximum number of 1's and *'s in any column, over all column permutations, therefore corresponds to minimizing the number of tracks and hence the area utilized in circuit realization.

Gate matrix layout is \mathcal{NP} -complete [9]. Despite many years of study, however, no algorithm is known to approximate it to within a multiplicative constant. Nor can it be approximated to within an additive constant, unless $\mathcal{P} = \mathcal{NP}$ [2]. In order to apply Theorem 1, our first task is to transform an arbitrary instance into a graph. To accomplish this, we expand a given matrix in the following manner. We replace any column with more than two 1's with a set of columns, each with only two 1's, representing all the possible ways to choose two 1's from that column. Next, we derive from this expanded matrix a finite simple graph, where we treat rows as vertices and columns as edges. Proofs for the next three results are rather tedious.

Lemma 1. [4] *Matrix expansion does not affect the cost of a gate matrix layout solution.*

Lemma 2. [4] *For every fixed k , the “yes” family of derived graphs is minor closed.*

Lemma 3. [4] *For every fixed k , the “no” family of derived graphs contains a planar element.*

So we apply the three preceding lemmas, fortify them with Theorem 1, and now bring on the armchair.

Theorem 3. [4] *For every fixed k , gate matrix layout is solvable in polynomial time.*

We went on to find quite an assortment of well-known layout problems for which we could prove analogs of Theorem 3. Just a short list would include disk dimension, minimum cut linear arrangement, topological bandwidth, crossing number, maximum leaf spanning tree, search number and two dimensional grid load factor. See [7].

Thus, “trolling” for applications using this general approach became for us a pretty standard recipe:

- look around for a provably difficult problem,
- fix parameter(s) as required,
- make sure it can’t now be solved by brute force or table lookup,
- devise problem transmogrifications as needed, and
- check for minor or immersion closure (an armchair is highly recommended at this step).

This line of work was really quite addictive. When our quest was successful, it usually turned out that the “yes” family was the one that was closed. But there were exceptions. Consider, for example, longest path. For every fixed k , the “no” family is actually minor closed. Moreover, most of the time obstructions were highly elusive, requiring enormous effort for comprehensive analysis (see, for example, [11]). But again there were exceptions. And again consider longest path. For every fixed k , the only obstruction to “no” family membership is a path of length k .

8 What the Hell Is VLSI?

At about this time Mike and I wrote several proposals, some successful and some not. One of our early successes was at the National Science Foundation, where we proposed to study the application of Theorem 1 along with other novel methods to combinatorial problems of relevance to VLSI design (hence our initial work

on gate matrix layout). There the current program director, Bob Grafton⁴, was most helpful and quick to sense the potential of these remarkable tools. After describing them to him over the phone, I liked his reactions so much that I more or less paraphrased one of his responses in our proposal abstract. Here is that abstract, from late 1985, with emphasis added to the sentence that I think best reflects Bob's statement to me:

*In the design and manufacturing of Very-Large-Scale Integrated (VLSI) systems, practical problems are characterized by fixed-parameter instances. For example, a parameter might represent the number of tracks permitted on a chip, the number of memory cells available, the number of processing elements to be employed, or other variables significant to the solution of the problem at hand. **In fixing the value of such parameters, we focus on the physically realizable nature of the system rather than on the purely abstract aspects of the model.** In this investigation, research efforts are concentrated on this central, practical feature of real VLSI design problems, whose domains span the spectrum from the gate to the systems architecture levels. Powerful and, in many cases, emergent techniques from the fields of complexity, graph and group theory are brought to bear on these fixed-parameter problems so as to yield exact or guaranteed approximate solutions.*

Observe from this that we were employing terms such as “fixed-parameter instances” and “fixed-parameter problems” well over a dozen years before fixed-parameter tractability had been formalized, codified and systematically presented [3] by Mike along with Rod Downey, who is also slated to contribute to this festschrift.

By this time Mike (and I as well, probably) had become much more caught up in the revolutionary algorithmic techniques than in the circuit problems themselves. So much so that by the time I learned that this particular proposal was funded and told Mike, he looked at me excitedly and said⁵ “Wow that’s really great! But what the hell is VLSI?” This was surely a curious remark from a co-PI on a proposal written to perform research on, drum roll please, VLSI. In fairness, we had probably not discussed that proposal since we had written it several months earlier. And I knew of course what Mike meant by his statement. The way he said it, however, made us both laugh out loud. Mike has always been singularly quotable.

9 Constructive Complexity

From the beginning, Mike and I recognized the need to address the show-stopping shortcomings of applications stemming from remarkable results such as Theorem

⁴ We are indebted both to Bob Grafton (NSF) and to Ralph Wachter (ONR), and of course to their anonymous review panels, for their early feedback, understanding and recognition of the long-term potential for what eventually has become the field of parameterized complexity.

⁵ I believe this quote is verbatim. It seems like only yesterday.

1. For one thing, the algorithms had staggering constants of proportionality. A little thought experiment may be useful here. Pick your favorite enormous constant such as, let's say, some reasonable upper bound on the number of fish in the sea. As it turns out, that's too small. So pick, say, a bound on the number of grains of sand on all the world's beaches. That's still much too small. All right then, let's try a bound on the number of elementary particle interactions that could possibly have occurred in the lifetime of the known universe. That's still too small! Those sorts of huge numbers are rapidly dwarfed by the "towers of two" constants employed in the algorithms of Robertson and Seymour. In this context, I am often reminded of a relevant statement perhaps dubiously attributed to Disraeli: "There are three kinds of lies: lies, damned lies, and statistics." Around this time I began pointing out to my graduate students that there are in fact at least four kinds of lies: lies, damned lies, statistics, and the big "oh" notation. After all, what does it really mean to claim you have, say, even a linear-time algorithm when its constant of proportionality is so outrageous? Mike and I subsequently worked for quite some time on a variety of solutions for this issue. We were generally able to mitigate constants greatly (plus reduce degree bounds), mainly through the use of graph width metrics.

Another weakness of applications based solely on Theorem 1 was the lack of any techniques for search or optimization. This theorem and its subsequent improvements provide algorithms for decision only. So Mike and I worked on this too. We were generally (but interestingly, not always) able to reduce search and optimization to decision, mainly through self reduction. These results are discussed in detail in [6].

A third shortcoming, an egregious one, and the only algorithmic deficiency I'll address in detail here, was non-constructivity. Theorem 1 and its analogs provide no general means for finding (or even recognizing) the promised algorithms. All we are told is that such algorithms must exist. So Mike and I set out to find ways to remedy this situation. How could inherently non-constructive tools ever be made constructive? It was a puzzler!

After many false starts, the constructivization methods we finally devised operate in a rather counterintuitive, and perhaps even paradoxical, fashion. We were able to prove that we could, in principle, write down an algorithm. We could show that it was correct. We knew it relied on the finiteness of its obstruction set. And we could watch it run as long as we liked. Yet we also could show that we could never know for sure the obstruction set itself and, sometimes, we could not even know an exact bound on the algorithm's running time.

The following is a greatly simplified version of a much more general result from [6], where we deal with arbitrary well-quasi-orders. I have stripped it down and restricted it here; otherwise I would need to introduce several fairly cumbersome definitions.

Theorem 4. [6] *Let F denote a closed family in the minor or immersion order. If the following are available*

1. a solution checking algorithm that runs in $O(T_1(n))$ time,
2. order tests that need at most $O(T_2(n))$ time, and
3. a self-reduction bounded by $O(T_3(n))$ time,

then $O(\max\{T_1(n), T_2(n) * T_3(n)\})$ time is sufficient to solve the decision version of F -membership.

I should probably explicate with an example. So let me select one familiar to most readers, and turn again to vertex cover. As previously observed, the “yes” family for any fixed k is minor closed. First, if a putative solution is proffered, its correctness can be checked in at most quadratic time. All one has to do is delete the vertices in the supposed cover and look to see if the resulting graph is edgeless. Second, we know that this family excludes a planar graph. We can therefore test whether an arbitrary input graph contains a fixed obstruction in at most $O(n \log n)$ time via a bounded treewidth argument [6]. And third, if a decision oracle reports that an input is a “yes” instance, we can self-reduce to a solution in a variety of ways. For example, we can iteratively eliminate each vertex v in turn, and re-invoke the decision oracle for $k - 1$. If and only if this oracle says “yes” do we mark v as a member of a satisfying cover and decrement k by 1. Thus, at most a linear number of self-reduction calls⁶ are required. Theorem 4 therefore guarantees that the entire procedure can be accomplished in $O(\max\{n^2, (n \log n) * n\}) = O(n^2 \log n)$ time.

The vast majority of problems known to be amenable to Theorem 1 are also amenable to Theorem 4. In fact, Theorem 4 generally gives us not just a constructively known algorithm, but a constructively known polynomial-time algorithm. This is because polynomial-time checking usually comes from membership in \mathcal{NP} , fast tests are available for both orders, and self-reduction is most always possible. Only those problems otherwise not even known to be decidable seem resistant to this constructivization. Intriguingly, the algorithm guaranteed by Theorem 4 relies on the correctness and finiteness of F ’s obstruction set — but we can never use it to learn the set! This is because no finite amount of observation will ever tell us whether the algorithm has found all the set’s elements.

Well that seems rather curious. What about just computing the set directly somehow? Mike and I thought a lot about that. We were only able to show that if one is given an algorithm for minor-closed F -membership in the form of a Turing machine, then there could be no algorithm to find all the relevant obstructions [6]. I’m sure the same is true for the immersion order. But of course that’s not very surprising. Precious few things are computable when arguments are reduced to the Turing machine model.

Not long after Theorem 4 became fairly well known within the community, Vijaya Ramachandran kindly pointed us to an intriguing idea generally attributed to Leonid Levin [13]. We were immediately attracted to Levin’s strategy, and sought ways to employ it. Soon enough we managed to produce the following

⁶ It is important that we consult decision oracles only for non-increasing values of k , and that we do not overly inflate the size of the graph as we modify it during self-reduction. These are known as uniformity and honesty requirements, respectively.

result, which I would probably characterize as illuminating and entertaining, but wildly impractical and unimplementable. Once again this is a simplified version of a more general result from [6].

Theorem 5. [6] *Let F denote a closed family in the minor or immersion order. And let $T_0(n)$ denote the time complexity of any algorithm solving the search version of F -membership. If the following are available*

1. *a solution checking algorithm that runs in $O(T_1(n))$ time,*
2. *order tests that need at most $O(T_2(n))$ time, and*
3. *a self-reduction (its time requirements are irrelevant),*

*then $O(\max\{T_0(n) + T_1(n) * \log T_0(n), T_2(n)\})$ time is sufficient to solve the search version of F -membership.*

Under reasonable assumptions about $T_0(n)$, $T_1(n)$ and $T_2(n)$, the algorithm of Theorem 5 is asymptotically optimal — yet we may never know exactly what sort of upper limit optimality provides. This is because we are only ensured a runtime that's bounded above by a multiplicative constant of any satisfying $T_0(n)$. We hugely exploit the fact that each procedure has a constant index in any fixed enumeration of all algorithms. But this gives us no systematic means for learning anything useful about how an algorithm achieving a low order $T_0(n)$ actually works.

I think constructivizations such as these remain a bit unsettling, even to this day. Theorem 4 tells us that we can, usually, know an algorithm. We can write it down. We can trace it as it employs a growing but finite obstruction set. We are assured that it will not loop forever. Nevertheless, no matter how long we watch it, we may never know the entire obstruction set upon which it relies. Similarly, Theorem 5 tells us that we can, in principle, construct an asymptotically fastest algorithm. We know exactly how it works. We can check its results. Yet we will never know its time complexity. What an odd turn of events. All this reminds me again of the fourth type of lie I mentioned earlier. Accordingly, in almost any practical sense, I would not argue against the sentiment that much of what we had done, especially in Theorem 5, was play a bunch of clever but dirty tricks with the big “oh.”

And so it went. Every time Mike and I found ways to remove various forms of non-constructivity from one arena, they would quickly seem to pop up in another. It was as if we were playing Whack a Mole at the complexity theory arcade. Despite all this, I think Theorems 4 and 5 (or rather their more general versions) and others that Mike and I devised have been big hits for many years. I am often asked about them. They seem to get at the core of what can and cannot be made constructive using well-quasi order theory. Working on them was a lot of fun for the two of us. They exemplify the sorts of results we obtained on what then were challenging algorithmic paradigms in a completely new research domain.

10 Community Reactions

Early on, I think only a few of our colleagues expressed much interest in what Mike and I were investigating. Fewer still gave us much in the way of encouragement. Notable exceptions include Gene Lawler, Vic Klee, Manny Blum, Steve Mahaney and a handful others (apologies to those whose names I've failed to recall). To them we are eternally grateful. We sorely needed the occasional attaboy, and they came through for us.

A few who shall remain nameless, on the other hand, found our earliest work somewhat blasphemous. In some quarters it took years for us to gain credibility. Depending on whom you asked, we were either the village idiots or the lunatic fringe. The specter of non-constructive algorithms was, I think, viewed as a bit of a threat by some who had staked their careers on the theory of \mathcal{NP} -completeness. You see, the usual response to any worry that \mathcal{P} might somehow equal \mathcal{NP} was that all those published and highly cited completeness reductions would still be important. They would simply become useful polynomial-time algorithms. But what if there were a non-constructive proof of, gasp, $\mathcal{P} = \mathcal{NP}$? After all, the standard approaches had long failed to resolve the question. Something radical might be required. In that case, all those swell completeness proofs would be meaningless. You could still map all of \mathcal{NP} to some \mathcal{NP} -complete problem Π all right, solvable in polynomial time via Theorem 1. But without constructive knowledge of the polynomial-time algorithm for Π , you're stuck. Of course today, most of us would probably bet that surely $\mathcal{P} \neq \mathcal{NP}$. Probably the same was true back then too, although word on the street was that hopelessly flawed proofs in both directions were running about 50-50. It is fairly remarkable that the question remains open to this day.

Some in our community even dismissed our early work as mere "mathematical curiosities." Perhaps you have heard stories from Mike about the disingenuous "elevator chat" at STOC, or seen the insulting "minor results" article actually produced in print. Although I would prefer not to name any names here, I was there for these and several other early snubs, sometimes poorly disguised as humor. It seemed to us that hubris among the theoretical computer science intelligentsia was never in short supply back then. While all has long been forgiven, it hasn't been forgotten. Perhaps I merely flatter myself, but I would like to think I have a pretty thick skin. So this was to me just part of doing business in computer science. Besides, I'm not a dyed-in-the-wool theorist, nor really very much of a pure theorist at all. But these early affronts by the in-crowd were, I believe, actually quite painful for Mike. They became huge and long-lasting influencing factors on his attitude toward research in general and theoretical computer science in particular. They helped shape his thinking, created in him something of a firebrand for the cause against elitism, and released him from the parochial and highly inbred model of theory as then practiced in the US. Over time, I think they helped steel his resolve to make his work truly world class and now so well respected around the globe. *Per adversas res fortitude!*

11 Shifting Gears

Thus, the early years were absorbed mainly with the positive side of the equation. We focused on concrete problems, speed-ups, self-reductions, constructivity issues and the like. While we brought forward the basic notions of non-uniform complexity and what is now known as fixed-parameter tractability, at that time we could only struggle with a variety of feeble ideas for lower bounds. As time marched on, Mike and I both took positions elsewhere. I stayed in the US, moving to the University of Tennessee and focusing mostly on applications. After stints at the Universities of New Mexico and Idaho, Mike eventually moved outside the US, first to Canada, later to slots in New Zealand and other countries, and finally to Australia, where he now holds a professorship at Charles Darwin University. During all those years and across all those moves, Mike has built up hugely successful collaborations with a coterie of renowned scientists. One of the things that rather quickly came out of all that effort has been the much-heralded negative side of the equation (by that I mean hardness and the \mathcal{W} hierarchy). Other sections of this festschrift will no doubt describe some of those collaborations.

12 Retrospective

When I first met Mike I was working on a wide variety of topics (as I still do today). At that time I was investigating problems in packing and scheduling theory, parallel computation, time-space optimal algorithms and several other areas. Maybe I have trouble committing myself just to one subject. In any case, I think pursuing multiple research interests can make it somewhat easier to find good collaborators, attract funding, and pique the interest of capable students. Plus I find it to be a lot of fun. On the other hand, spreading your time across multiple target areas is not really the best way to build brand recognition for your research program. Instead, it's probably a lot smarter to pick a single focus area and concentrate on being the world's best in that one area. (Young scientists take note!) And this is where Mike has always excelled. I think I worked pretty hard in those early days, but again I had several ongoing research projects. In contrast, Mike worked almost all the time and single-mindedly, with minimal distraction, on what eventually evolved into parameterized complexity. Except for an occasional sideline interest (such as Computer Science Unplugged), he generally still displays that fantastically keen focus today.

So I've watched Mike grow from being the voice of "What the hell is VLSI?" to his now being the voice for a whole new computational discipline. From the day his house self destructed when its pipes froze then re-thawed, to his now being in such international demand that he needs no permanent house at all. Mike and I have worked, played and laughed together for over a quarter century. Friends and colleagues like that don't grow on trees.

Let me therefore say congratulations, Mike, old friend, on the completion of your 60th year on Planet Earth. I wish you continued good health, much happiness, and many many more fruitful and productive years to come!

Acknowledgments and Disclaimers. I would like to take this opportunity to thank Rod Downey and Daniel Marx for their helpful suggestions in improving the presentation of this collection of reflections on Mike's career. In fact I would like to thank all the festschrift organizers for inviting me to prepare this contribution. At their behest, I have done my dead level best to describe the background and genesis of fixed-parameter tractability. Thanks also go to Fran Rosamond. Her enthusiasm, spontaneity and consistent good humor are positively contagious.

The opinions expressed here are mine alone, and should be foisted upon neither Mike nor the organizers of this project. I have tried to be open and forthright, yet politic, calling events honestly as I recall them. I have pulled only a few punches, and those mostly in the uneasy early relationship Mike and I had with the computer science theory establishment. I would also like to thank profusely the many who helped inspire Mike and me along the way, and apologize again to those whose names I've inadvertently failed to mention.

References

1. Chen, J., Kanj, I.A., Xia, G.: Simplicity is beauty: Improved upper bounds for vertex cover. Technical Report TR05-008, DePaul University, Chicago, Illinois (2005)
2. Deo, N., Krishnamoorthy, M.S., Langston, M.A.: Exact and approximate solutions for the gate matrix layout problem. *IEEE Transactions on Computer Aided Design* 6, 79–84 (1987)
3. Downey, R.G., Fellows, M.R.: *Parameterized Complexity*. Springer (1999)
4. Fellows, M.R., Langston, M.A.: Nonconstructive advances in polynomial-time complexity. *Information Processing Letters* 26, 157–162 (1987)
5. Fellows, M.R., Langston, M.A.: Nonconstructive tools for proving polynomial-time decidability. *Journal of the ACM* 35, 727–739 (1988)
6. Fellows, M.R., Langston, M.A.: On search, decision and the efficiency of polynomial-time algorithms. In: *Proceedings of ACM Symposium on Theory of Computing*, pp. 501–512 (1989)
7. Fellows, M.R., Langston, M.A.: On well-partial-order theory and its application to combinatorial problems of VLSI design. *SIAM Journal on Discrete Mathematics* 5, 117–126 (1992)
8. Garey, M.R., Johnson, D.S.: *Computers and Intractability: A Guide to the Theory of \mathcal{NP} -Completeness*. W.H. Freeman and Company, New York (1979)
9. Kashiwabara, T., Fujisawa, T.: An \mathcal{NP} -complete problem on interval graphs. In: *Proceedings of IEEE Symposium on Circuits and Systems*, pp. 657–660 (1979)
10. Kinnnersley, N.G.: The vertex separation of a graph equals its path-width. *Information Processing Letters* 42, 345–350 (1992)
11. Kinnnersley, N.G., Langston, M.A.: Obstruction set isolation for the gate matrix layout problem. *Discrete Applied Mathematics* 54, 169–213 (1994)
12. Kuratowski, K.: Sur le problème des courbes gaushes en topologie. *Fundamenta Mathematicae (French)* 15, 271–283 (1930)

13. Levin, L.A.: Universal enumeration problems. *Problemic Peredaci Informacii* (Russian) 3, 115–116 (1972)
14. Lopez, A.D., Law, H.-F.S.: A dense gate matrix layout method for MOS VLSI. *IEEE Transactions on Electron Devices* 27, 1671–1675 (1980)
15. Robertson, N., Seymour, P.D.: Disjoint paths - a survey. *Journal of Algebraic and Discrete Methods* 6, 300–305 (1985)
16. Robertson, N., Seymour, P.D.: Graph minors IV. Tree-width and well-quasi-ordering. *Journal of Combinatorial Theory, Series B* 48, 227–254 (1990)
17. Robertson, N., Seymour, P.D.: Graph minors XIII. The disjoint paths problem. *Journal of Combinatorial Theory, Series B* 63, 65–110 (1995)
18. Robertson, N., Seymour, P.D.: Graph minors XX. Wagner’s conjecture. *Journal of Combinatorial Theory, Series B* 92, 325–357 (2004)

The Birth and Early Years of Parameterized Complexity*

Rod Downey

¹ School of Mathematics, Statistics and Operations Research
Victoria University

P.O. Box 600, Wellington, New Zealand

² Isaac Newton Institute for Mathematical Sciences
20 Clarkson Road Cambridge CB3 0EH, United Kingdom
rod.downey@vuw.ac.nz

Abstract. Through the hazy lens of (my) memory, I will try to reconstruct how Mike Fellows and I, together with some co-authors in some cases, came up with the basic papers in parameterized complexity.

1 Introduction

When I agreed to do this archaeological exercise, I was rather enthusiastic until I tried to remember dates and places and what we did and where. Through cunning questions of Mike, without disclosing the existence of the Festschrift, I have confirmed many of the “facts” below. Also, I have e-mails from December 1990, and letters from 1991 and 1990. Unfortunately, many discuss “following our phone conversation” so in many cases it’s back to memory.

I will not try to say who did what, since mostly it was a happy coalition of two friends. Anyway, my policy has always been that “joint work” is joint work. In our case it was definitely so : a nice interaction of complementary skills. Only as an illustration of this complementarity, I will say what happened with the very first papers [DF91, DF92a, DF92b]. It is both an interesting study in serendipity and an interesting study in complementary skills.

Before I begin, I take this opportunity to salute my longtime friend, my favourite co-author, and certainly my only co-author who understands the value of surfing. Congratulations Mike on the occasion of your 60th.

2 Beginnings

I know to within 2-3 days when I first met Mike Fellows. It was at the ACCMCC (a combinatorics conference series in Australia and New Zealand) conference at Massey University. This conference was in a medium sized rural town in New Zealand called Palmerston North, December 3-7, 1990.

* Research supported by the Marsden Fund of New Zealand. Thanks to Mike Langston and Moshe Vardi for some helpful comments and corrections.

Palmerston North is not the most exciting place in the world. In fact, much to the ire of the locals in 2006, John Cleese (of Monty Python fame) was quoted as saying “If you ever do want to kill yourself, but lack the courage, I think a visit to Palmerston North will do the trick.¹” You may then ask yourself why Mike was visiting New Zealand for this meeting? As it turns out, in his youth he had seen a famous surfing movie called “The Endless Summer” and it featured surfing in the apparently exotic location of New Zealand, particularly Raglan and Shipwreck Bay². He hoped to combine a visit here for the conference with a visit to some of the local surfing locations. But here’s a classic case of serendipity. I did not have too much interest in ACCMCC, but in those days there was almost no research travel money to be had, certainly not by mathematicians or computer scientists in New Zealand. Hence, if there was any conference near, you went to it. My attitude was (and remains) “Who knows, maybe you might pick up an idea or two”. In those days, you might only get one chance a year³.

I can pretty well say that I met Mike on the 4th December. I know it was the day of the conference dinner and Mike was speaking in the afternoon. After his talk, I recall meeting him outside of the seminar room, saying that I thought the material was very interesting and loved the talk. I said it reminded me of a paper I had read recently from a *Contemporary Mathematics* proceedings.

Mike said something to the effect that “it should remind me as *he* wrote that paper and this is what it was about” (Fellows [Fe89]). It’s always wonderful to run across someone who has actually read one of your papers, even if they don’t recall who wrote it, and perhaps this was all to the good.

¹ See, for example, <http://www.abc.net.au/am/content/2006/s1586512.htm>. The silly part was that the local tourist board took it all seriously and then the locals promptly called the local rubbish tip “Mount Cleese.”

² My understanding is that on this trip the film gave a false impression, in that everywhere Mike went was blown out and cold.

³ This was all before the Marsden Fund for basic science was set up in New Zealand, under the initiative of Sir Ian Axford. In those halcyon days, I recall being ambitious and hungry for the most recent ideas. There were a few small grants available. I did get a seeding grant from a committee called ISAT enabling me to have one overseas trip and several American mathematicians to visit me here in New Zealand. I am forever grateful for that. To get overseas, I would write to many universities in the US, asking would they be so kind as to give me a little money if I gave a talk, and then cobble the trip together using money from giving seminars at maybe 6-7 universities. Once I asked the travel agent to get me to Haifa the cheapest way he possibly could: Wellington-Sydney-Tokyo-Amsterdam-Tel Aviv with bad connections, 54 hours. (KLM had a deal for a \$50 to anywhere in “Europe” side.)

There were some grants available to people via what was called the “University Grants Committee” but any proposal needed to get out of the local university, in those days here controlled by non-mathematicians, who everyone knew were “world class” by some unknown mysterious mechanism. I once tried to get one of these grants to visit Mike Fellows for collaborative research, and received the official reply that we don’t support “Research Fellows.” This response showed how seriously they read the proposal. Fortunately things changed once all proposals were internationally vetted.

Mike started telling me about his work with Langston [FL87, FL88] of using Robertson-Seymour (e.g. [RS86a]) methods to demonstrate polynomial time complexity non-constructively. Mike asked me about my background. Hearing it was in logic and computability theory/structural complexity he suggested a problem I might be interested in. He gave me a copy of his paper Abrahamson, Ellis, Fellows and Mata [AEFM89] the “PGT” (“polynomial generator tester”) paper. The contents of the PGT paper are quite important for this story and I will soon discuss them in detail. We decided to sit together at the conference dinner and discovered that we also shared a love of surfing and a love of wine, having had a couple of bottles of some Villa Maria Cabernet Merlot 1989. Mike suggested I might like to work with him on trying to prove a Ladner style density argument for the PGT setting for the complexity classes for “families of relations.”

Ladner’s Theorem [La75] is often quoted as being that if $P \neq NP$ then there is an “intermediate” NP language, which is neither NP-complete, nor in P. But in fact, Ladner proved that if A and B are computable languages with $A <_T^P B$ then there is a computable language C with $A <_T^P C <_T^B$. In unpublished work (a proof can be found in Downey and Fortnow [DFo03]), Shinoda and Slaman proved that you can remove the hypothesis that A and B are computable, and that this result holds for any languages $A <_T^P B$.

Since it is relevant to the story at hand, I will digress as to how I got into complexity theory at all. My background was in computable algebra and classical computability theory. Here one calibrates the universe into classes of relative computability under various reducibilities and in applications to algebra, you look at things like, e.g. computable fields and ask if every computable field has a computable algebraic closure. As with all things I study, I retain an interest and recently showed that computing whether two finitely presented groups have the same first three terms in their integral homology sequence is what is called Σ_1^1 complete. This shows that as an invariant it is as bad as it can possibly be. (Downey and Montalbán [DM08].)

At a certain point in the late 1980’s, I resolved to learn some complexity theory, as it is always interesting to work in new arenas. This is something Mike is extremely keen on. He reads a lot. He can *see* many elegant applications into new areas of, for example, biology, VLSI design, linguistics, learning theory etc⁴. Branching out into a new area is something you can do at a small department like mine, and I did this by teaching complexity theory (as per Garey and Johnson [GJ79], and Balcazar, Diaz and Gabarro [BDG87]). I wrote some papers on classical “structural complexity” such as the structure of the polynomial-time degrees (for example [DGHM89]).

Because of this background, I was familiar with Ladner’s results so I guess I was a good candidate for Mike’s question.

⁴ One thing I always liked in Mike’s homes was that there were always teetering piles of books from all kinds of areas of human endeavour, and papers from many areas of science scattered everywhere.

The point of all of this was that Mike left me a copy of the PGT paper, and I promised to look at it as soon as I could. I began the next day.

It had long been recognized that some kinds of intractability might be better than others as mentioned specifically in Garey and Johnson [GJ79]. There had been no real quantification of what this meant, except perhaps forays into average case complexity, approximation algorithms and strong NP-completeness. There seemed no history of worrying about parameters in the input. Whilst it seems that earlier authors such as Ken Regan in [Re89] (who talked about k -DOMINATING SET in passing) and Moshe Vardi [Va82]. Vardi pointed out that the input for database-query evaluation consists of two components, query and database. For first-order queries, query evaluation is PSPACE-complete, and for fixpoint queries it is EXPTIME-complete, but, if you fix the query, the complexity goes down to LOGSPACE and PTIME correspondingly. In particular, the size of the database was not the right complexity for database query complexity and the size of parameter counted. Also in the 80's were the papers Vardi and Wolper [VW86] and Lichtenstein and Pnueli [LP85] who pointed out that the input for LTL model checking consists of two components, formula and transition system. LTL model checking is PSPACE-complete, but if you fix the formula, the complexity goes down to NLOGSPACE.

So people in the database community were very aware that fixing a parameter makes an intractable problem tractable. In retrospect the key is that they did miss the *big* difference between query evaluation and model checking. In query evaluation the dependence on the formula is exponential, while in model checking it is multiplicative. Indeed, as was shown later, model checking is FPT and query evaluation is likely not FPT. All this happened in the 1980s, but in spite of the clues, these workers completely missed parameterized complexity theory. Of course other earlier work is discussed in Mike Langston's article in this volume.

The first real breakthrough was the PGT paper of Abrahamson, Ellis, Fellows, and Mata [AEFM89]. In [AEFM89], they modeled parameterized problem by what they called *polynomially indexed relations*. The crude notion of (nonuniform) FPT is definitely in [AEFM89] as the notion of easiness. Specifically, Abrahamson, Ellis, Fellows and Mata defined a P -indexed family of relations $\Pi \subset \Sigma^* \times \Sigma^*$ with the n -th slice having $|x| = n$. Then this class is said to be⁵

- P -bounded if there is a polynomial p such that $(x, y) \in \Pi$ implies $|y| \leq p(|x|)$.
- P -checkable meaning that there is a polynomial time algorithm to decide if $(x, y) \in \Pi$, and
- P -indexed meaning that there are polynomial time algorithms to compute
 - (a) $(1^n, i) \mapsto y$ such that $i_n(y) = i$ and
 - (b) $(1^n, y) \mapsto i_n(y)$ for $y \in \text{ra}(\Pi_n)$.

Then the problem kinds they considered (“PGT=polynomially generator tester pairs”) were of the form as follows.

⁵ The reader should not stress too much about the details of the definition, I only give it for historical interest, and to show that it is extremely unwieldy.

Input. $x \in \Sigma^*$ and $j \in \mathbb{N}$ with $j \leq q(|x|)$.

Question. Is there a y with $(x, y) \in \Pi$ and $i_{|x|}(y) \leq j$?

The example was VERTEX COVER. In this [AEFM89] setting, deciding whether there is a vertex cover of size $\leq k$ corresponds to the input (G, J) the q bounded search problem for $\Pi = VC$ and $j = j(n) = \sum_{i=0}^k \binom{n}{i}$. (This is quoted straight out of [AEFM89].)

There is a similar multi-line definition of a many-one reduction from Π to Π' as a function $f : (x, i) \mapsto (x', i')$ and

- f is computable in time polynomial in $|x|$ and $\log i$.
- There are polynomials r, s, t with $|x| \leq r(|x'|)$, $|x'| \leq s(|x|)$, and $i' \leq t(i)$.
- $\exists x((x, y) \in \Pi$ and $i_{|x|}(y) \leq i)$ iff $\exists y'((x', y') \in \Pi'$ and $i'_{|x'|}(y') \leq i')$.

Using this very cumbersome definition, Abrahamson, Ellis, Fellows and Mata showed that things like k -LINEAR INEQUALITIES had the same complexity as k -SHORT SAT (for arbitrary formulas), and asked the question of how things like k -DOMINATING SET fell into this classification. (It does not seem to as the classes are really somehow concerned with nonuniform $W[P]$.) Most of the reductions, not surprisingly, look akin to recycled LOGSPACE completeness results, and all resurface later in our paper [ADF95] with Abrahamson where we look at $W[P]$, and alternation. There is an error in the paper as it purports to show that weighted sat is (essentially) $W[P]$ complete, where in the proof of Theorem 6.5 it is claimed that unravelling of the formula is equivalent to asking if it is satisfiable, (which is true) but the point is that the process makes more variables true so the reduction cannot be parametric. However, we were able to rescue a lot of results in [ADF95], and discover a further collection of $W[P]$ -complete problems (this happened in 1993).

As you all know, Mike's personal background is pretty colourful. It involves the armed forces, jumping out of planes in parachutes, later fishing in dangerous waters, colourful living in San Diego and then finally getting into graduate school at UCSD due to the intervention of Michael Freedman of Fields Medal fame⁶. I think that this eclectic background has enabled him to be very creative, and I think you can see this in the [AEFM89] paper, as well as his earlier work with Langston [FL87, FL88] (article included in this volume). Both he and I are from relatively modest childhoods, and I believe that this often makes you quite ambitious. Maybe this is where the initial drive came from.

The story continues as follows⁷. I sat and stared at the [AEFM89] paper for a few days and thought two things. First, there is the kernel (no pun intended) of a *very exciting idea* in this paper buried under layers of ugliness, which does not get to the key issue; and second I will have a lot of trouble proving a Ladner Theorem with such an unwieldy definition. *So maybe there is a simpler formulation.*

At a certain moment I recall thinking why not simply study languages $L \subseteq \Sigma^* \times \Sigma^*$ or $\Sigma^* \times \mathbb{N}$, and have reductions as what we now see as parametric connections $(x, k) \mapsto (x', k')$. The rest of the [AEFM89] hides the core issue.

⁶ The occasion prompting Freedman to intervene being present in one of Mike's plays.

⁷ This is the only point I will say who did what.

I am sure that this all came out of my logic training. I then figured out the density theorem (eventually being published in [DF93] and announced in the abstract with Karl Abrahamson [ADF93]) and wrote and maybe faxed Mike several letters including a 26 page handwritten letter with a lot of proofs. We were very excited and we spoke on the phone for long periods. Since our mathematical backgrounds are quite distinct and Mike had not seen priority arguments before, likely he had trouble following my scrawls. *But* after we talked through the issues on the phone and he clearly saw many ways in which the simple definitions could be applied to DOMINATING SET. It must have been by the 18th December 1990 as I have an e-mail where Mike summarizes all the definitions. In that e-mail he sets down his ideas towards using logical depth, in the format of weft, (and hence the W -hierarchy) as the basis of classifying parameterized complexity. So you can see this all happened very quickly, maybe a couple of weeks. I thought weft was a terrific idea but had seen no proofs.

Where did Mike get this nice idea from? Certainly he had been thinking about DOMINATING SET and INDEPENDENT SET as well as VERTEX COVER. If you look at the logical form of the first two, you can see the form of $W[2]$ and $W[1]$ respectively, provided that you are prepared to use boolean circuits as a platform for the intractability “core problems”. Only later with Liming and Jain were we able to get nondeterministic Turing machines into the picture. ([CCDF96].)

Even now we don’t have Turing machines in the picture for the miniatures $M[1]$, and this is a great open question. (Is $k \log n$ Turing machine acceptance in $M[1]$?)

It is also interesting that at the bottom of the e-mail of the 18th December are the questions (i) if $P \neq NP$ does this hierarchy separate (ii) If $W[t] \neq W[t + 1]$ are there infinitely many equivalence classes, and (iii) Relationship between the $\cup W[t]$ and $W[P]$. At the time (i) has us thinking about oracle separations, (ii) relates to density and (iii) remains interesting even now.

At this stage, Mike had a lot of grant money, and asked me to visit him. He thought we could work well together and work out the details of this very attractive material. I was very excited as I could see that there was a lot of potential. Clearly I thought that what we were doing might be important, as I have kept some of these early e-mails⁸.

In early 1991, I visited Mike for the first time⁹. I stayed at Mike’s house in Victoria on Vancouver Island. I recall getting there and him telling me that they were practicing ecological front lawns (which seemed to equate to not mowing ever). The house was occupied by Mike’s family (two nice kids Max¹⁰ and Hanna)

⁸ Or the other interpretation might be that I am a horder of such things. Only in 1992 did I keep some kind of e-mail file deliberately.

⁹ Of course, one hero here was my poor wife Kristin who had to deal with two children under three by herself. On this trip I think she tripped carrying a baby car capsule, and sprained her knee.

¹⁰ One great memory I have of Max was that, on a later trip, we were going over to a place called Cox Bay on the west coast of Vancouver Island for a few days surfing and work. Mike said to Max we had to leave soon and was he packed? Max grabs a jacket, jumps in the car and says something to the effect of “Let’s go!” That was the trip where Mike forgot Roberta’s clothes, something we discovered half way there.

and his then wife Roberta. I recall that it usually smelled of coffee, and was covered with piles of books and papers, as I mentioned earlier. There I discovered that Mike would get irritable if he did not eat enough eggs, and we would eat a lot of Mexican food. I would stay in a room in the basement, next to bathroom that Mike “fixed” with a technique which inevitably involved vast amounts of silicone being squeezed in a large ungainly mess.

Mike essayed to me details of his ideas of using circuit weft as a basis of classifying parameterized complexities. In particular, Mike had the *key reduction* for CNF SAT, which is used throughout [DF92a, DF92b, DF95a]. During that visit, I well remember working each day on a little white-board or maybe flip-chart at his home where many of the details of the first paper(s) [DF91, DF92a, DF92b] were worked out. Already, Mike had the fundamental idea here for the $W[t]$ classes, and for $t \geq 2$ we figured out the details on this trip.

Aside from the project we talked about lots of other things ranging from poetry to mathematics education. I had been involved in mathematics education in New Zealand, once organizing a conference for teachers at my home university. I found this something that totally sucked up your energy. I gave Mike the “sage advice”, don’t do it, it will ruin your research. Well how wrong I was on both counts! I am glad that there are nice accounts of Mike’s initiatives with Neil Koblitiz (“The Mathematics Liberation Front”), Tim Bell and Nancy Casey.

Mike also knew of my interest in surfing as we had spoken of this in Palmerston North and on the telephone. I brought my wetsuit to Canada and we went to this place called Sombrio. In the early 1990’s this was a place occupied by a few guys who seemed to live like some kind of refugees from the 1960’s in old shacks. In those days, Sombrio was somewhat difficult to get to. A couple of hours drive, then a walk for half a kilometer through knee deep mud. Boy, was that all fun. In 2008, Mike and I went there for old times, when STOC/IWPEC was in Victoria. Now, it is all made up track, part of the West Coast Trail, and there were 50 guys there surfing!¹¹

As with all the trips, Mike and I spoke about maths most of the way. It’s a great model for research.

I have a draft manuscript ([DF91]), dated March 25, 1991 called “A Completeness Theory for Fixed Parameter Problems” which has the new definitions, weft ideas, some hardness proofs, and many of the basic results from [DF92a, DF92b]. It is not overly well written but has problems considered like k -PERFECT CODE, k -NOT ALL EQUAL SAT, k -CNF SAT, k -DOMINATING SET, k -INDEPENDENT DOMINATING SET, and the like. It also has a collection of FTP examples like FEEDBACK VERTEX SET, PLANAR FACE COVER NUMBER, MIN CUT LINEAR ARRANGEMENT, GRAPH GENUS, known mainly to Mike at that time. It has the basic weft reductions, though I think it slurs over the “obliviousness” of the k -CNF SAT reduction needed for the induction for the higher wefts. I also have

¹¹ On another trip, in 1992, we took Mike Hallett, one of Mike’s PhD students, and we put him in a dive suit. He fell off a wave and was washed across the reef (without touching it) looking like the gingerbread man as he could not bend too well in 9-18 mm of rubber.

some copies of some old slides of Mike's where he spoke at a meeting at Manitoba on this material a little later. (This was the basis of the paper [DF92a].) I recall that we submitted [DF91] to FOCS, and it was rejected¹².

It was quite early on that we noticed the issue of uniformity in the reductions. However, it was not really till the March that we refined this to the three definitions we now have. This occurred after a debate as to whether GRAPH GENUS was strongly uniformly FPT. GRAPH GENUS is only nonuniformly FPT on the face of it. Mike pointed out that it was uniformly FPT by Fellows and Langston [FL89b]. It was only shown to be strongly uniformly FPT in 1999 by Mohar [Mo99]. I think to some extent this shows where our "head space" was at the time, as we were still fascinated by the Robertson-Seymour material.

Incidentally, I still think that there is a very interesting project with this material. The hypothesis $FPT \neq W[1]$, say, has two very different meanings depending on whether we mean uniform or nonuniform. In the *nonuniform* case it says that determining whether a 3CNF formula has a weight k satisfying assignment is not in $DTIME(n^c)$ for some fixed c , and hence from some slice onwards, deterministic algorithmic must take more time. In the uniform case it is apparently weaker, and apparently it "could" be that all the slices are in $DTIME(n^c)$, but only nonuniformly. This is because the uniform case asks for a *single* algorithm to witness this inclusion. When I think of the issue I think that $W[1] \neq FPT$ *nonuniformly*, as the spirit of the programme.

3 Precursors

Before we move on, I guess we should look at where these ideas germinated. Yes, it is true that the relevant event was the [AEFM89] paper but also, in retrospect, you can see the ideas crystallizing out from earlier considerations; especially of the work of the two Mikes, Fellows and Langston, and their co-authors.

Even though we evolved to think of this as addressing practical computing, a real inspiration was the theorem of Robertson and Seymour which stated that finite graphs were well quasi-ordered under the minor relation, and immersions. Furthermore for a fixed graph H , $H \leq_{\text{minor}} G$ is $O(|G|^3)$. Hence any minor closed class had a polynomial time (FPT anyway) recognition algorithm, in spite of the fact that we did not know what it was. This is a stunning theorem, and it has yielded a revolution in graph theory in the last 30 years. The algorithms stemming from applications of Robertson-Seymour wqo theory are, or course, wildly impractical. Mike tells me of a famous computer scientist saying "This is not computer science, it is mathematical curiosity!" *But there are so many practical, nearly practical, and fascinating spin-offs.*

I have something else Mike gave me. An old proposal by him and Langston to use wqo methods to design VLSI circuits. And it was funded by the NSF,

¹² I have only ever been involved in two submissions to FOCS/STOC. One was [DF91], and the other was the recent one on kernel lower bounds [BDFH08]. Both were rejected.

and later the Office of Naval Research! Wow, is that blue-sky research. But look what it yielded : If nothing else, parameterized complexity.

The point here is maybe Mike was sensitized by his work with Langston (such as [FL87, FL88]) on applying and effectivizing Robertson-Seymour wqo theory. It focuses us on the issue of the parameter, *then* the algorithm once the parameter is known. In the Robertson Seymour case, the parameter is the obstruction set.

Also surely related here is the paper on cutset regularity (=finite index) by Fellows and Langston [FL89] that was quite important I believe. Fellows and Langston used an analogue of the Myhill-Nerode Theorem combined with a parsing language for graphs of bounded treewidth to establish a general methodology for fixed parameter tractability (where the parameter is treewidth) as per Courcelle's Theorem (as discussed here in Downey [Do12]), and a method of establishing that something is likely hard by showing that it is not of finite index.

Additionally, with the wisdom of hindsight, it spotlights the notion of implicit parameters like graph width metrics. For us, however, this material was more important when the book was being written.

On the other hand, in retrospect, it might have been a bit unfortunate to tie the FPT material to the Robertson-Seymour material when we spoke. Many listeners thought that what we were doing was basically applying Robertson-Seymour. For example, it is really striking how much it is mentioned in my Dagstuhl 1992 abstract. That is, initially, we failed to focus on practicality.

There are other precursors such as Kintala-Fischer's [KF80] model of limited nondeterminism, but we were unaware of this paper at the time. Also, Kintala-Fischer approach there does not split the problem into slices, is not applied anywhere, and has trouble dealing with the issues we deal with.

4 Figuring Out $W[1]$ and the Great Kiwi Road Trip

We found that we worked well together. Mike decided to visit me in Wellington. I cannot remember when, but it was almost certainly the northern summer, so maybe May or June 1991.

At the time, we had this framework (which was submitted to FOCS), lots of enthusiasm, and the whole of the world of NP-complete (and other) problems to see if we could find other interesting applications. The relative complexity of the proof of k -DOMINATING SET being $W[2]$ complete showed us that the reductions likely would not be easy.

In particular, we had not figured out the situation for $W[1]$. Given the relative delicacy of the reductions, we strongly believed that $W[1]$ would stratify into an infinite collection of levels $W[1, t]$. Here $W[1, t]$ denoted the class of problems *FPT*-reducible to depth 2 weft 1 circuits with one large *And* gate and whose small *Or* gates above have fanin bounded by t . We decided to make a surfing road trip and work on these issues, and my wife let us go.

Thus I grabbed my “Guide to New Zealand Surfing”, surfing gear, two books of poetry (Collected Poems by Michael Dransfield, and one called “Applestealers” about “new poetry” in Australia)¹³, Garey and Johnson, lots of paper, two clipboards and some preprints, and off we went. We basically circumnavigated the North island of New Zealand below Auckland, traveling to New Plymouth, Raglan, Mount Maunganui and Newdick’s Beach, Gisborne, Mahia, Napier, White Rock and then home. For about a week, we drove, worked, surfed and had a fantastic and incredibly productive time. Contrary to our intuition, we realized that there was *no* stratification of $W[1]$. This is the basis of the core paper [DF95b], and contains that lovely reduction for RED/BLUE NONBLOCKER and hence the completeness for CLIQUE and INDEPENDENT SET.

We also discussed another question brought up by Mike in an e-mail of February 27, 1991. Mike says he noticed this “weird thing”, which was that a certain problem whose unparameterized version was in Σ_2^P did not seem to fit the model we had. He said “Maybe the whole hierarchy is some kind of analog of the polynomial time hierarchy...” “Or maybe there is some kind of weird combinatorial reduction placing this above the current hierarchy.” I don’t recall that we made progress on this, but on my next visit to Mike in April or May 1992 we worked on this, eventually also with Karl Abrahamson, resulting in the paper [ADF95].

Like much of our time together, that first trip had a lot of great memories. The high point was coming over the hill at Mahia, and it looking like a surfing movie: lines stacked to the horizon. We surfed till we dropped and then drove that night into Napier. Mike slept most of the way. One the way in, we decided we needed a beer, and stumbled into this bar on the highway, not noticing the trucks and motor bikes outside. The place was full of large, tattooed and scary people, so after a quick beer we ran like rats.

5 Getting Published and Promoting the Material

Some time late in 1991 we received the news that the abstract [DF91] had been rejected from FOCS. We were really annoyed. Later people I spoke to like Stuart Kurtz and Lance Fortnow said that the abstract should have been accepted. I had been told by a number of people that this is why there are so many spin-off conferences like CCC as STOC/FOCS has strong opinions as to what constitutes an advance.¹⁴

Although it anticipates things somewhat, later we had reviews of some of our papers which said really revealing things like: “What this subject really needs is

¹³ Mike can read while in a moving car, and with me driving, we would alternate between a little poetry and a lot of maths.

¹⁴ Whilst I serve on many CS conference committees I don’t like the method. I always think of the art contests in Paris in the late 19th Century, and think of the paintings rejected; Van Gogh, Rembrandt, Rousseau, etc. Who do we remember now? I think that these institutions are intrinsically conservative. Moshe Vardi had a very interesting article about this in a recent Communications of the ACM. (Vardi [Va09].)

for it to be developed by someone like *here unnamed famous CS professor* and their students.” Hardly interacting with the *science* or *merit* of the work.

Having found out about the rejection, we decided to try for the 1992 *Conference on Computational Complexity* or *Structure in Complexity Theory* as it was then known. I have a version of the abstract we submitted dated December 9th 1991. Certainly it was much stronger than the earlier abstract, and includes the $W[1]$ collapse.

Mike was already doing what he does so well. Traveling around interacting and spreading ideas. He had had some excellent feedback both informally and about the material when he gave seminars.

The first time I spoke on this material was at Schloss Dagstuhl 9.00 am on Monday the 3rd of February 1992. I was the first speaker in the whole *Complexity Theory Seminar*. I was luckily invited for my early work in structural complexity. I had flown in from Wellington the day before. I could see in the eyes of the audience that this was a “good idea.” It does give me pleasure to go back and read the relevant abstract from the book they have at Dagstuhl.

For some naive reason I had expected lots of workers at that meeting to stop what they were doing and launch into the new theory. I particularly thought would happen this after Mike and Hans Bodlaender [BF95] showed that k -PROCESSOR SCHEDULING would likely not be in polynomial time, using our technology, *without* showing that it was NP-complete. More specifically, it had long been known that if k was part of the input, then k -PROCESSOR SCHEDULING is NP-complete, but for a fixed k , the NP-completeness or otherwise of k -PROCESSOR SCHEDULING is a prominent problem in the back of Garey and Johnson [GJ79]. What Hans and Mike did was to show that it is $W[2]$ -hard. This means that, assuming $FPT \neq W[2]$, there should be no feasible algorithm for large k . The hidden message of the Bodlaender-Fellows breakthrough is that it is possible to prove hardness without establishing NP-completeness¹⁵.

The mass parameterized migration did not happen with the exception of a few cases like Ken Regan. In fact, I think the majority of workers in complexity theory at the time remained and possibly remain unaware of the definitions, which is kind of a shame since it seems one of the few successful coping strategies.¹⁶ With the exception of some people such as Bill Gasarch, Alan Selman and Eric Allender have kept track. Even for those who seemed to like it, it was quite different from what they were doing at the time¹⁷.

¹⁵ Later this methodology was taken up by Aleknovich and Razborov [AR01] who recognized the value of complexity classes sensitive to issues within polynomial time.

¹⁶ Notably, most texts on computational complexity don’t even mention it. Perhaps this is because complexity theory sees itself as being concerned with showing something *can’t* be done, whereas a nice aspect of parameterized complexity is the focus on *trying to serve practical computation*. I am willing to believe $P \neq NP$, and as a consequence we need a complexity theory that “serves mankind” in the form of practitioners.

¹⁷ But at least both Mike and I were invited to the next *Complexity Theory* seminar at Dagstuhl in 1993.

In general, I think the majority of people keep doing what they do, but a little more on this later. At that Dagstuhl meeting the big thing was Ogiwara-Watanabe [OW91] and the leftset technique for looking at constrained reductions from sparse sets. The other thing was to define new complexity classes. There was an edict put out in this meeting that it should be illegal to define a complexity class and not populate it with a concrete problem. Fortunately for us ours had members!

Dagstuhl was very interesting in those days, as it was only the “castle” and did not include the lovely new “ring” building joined to the castle by the bridge. It had the worst Internet in the universe upstairs where the games rooms are now. I recall that loading a single page over the Internet via the dial-up modem would take an hour. I remember learning a lot about GRAPH ISOMORPHISM at that meeting. Some of us also went for a run as a group in some old tracks, with totally out of date maps and got totally lost. An hour run turned into a 2.5 hour one.

In 1992, I had a sabbatical and was to spend May–December at Cornell University. I recall dropping in to Mike on the way. I did not know much about treewidth nor about wqo theory, and Mike set about teaching me that. He showed me Bodlaender’s (and his student’s) work on algorithms for graphs of bounded treewidth and explained his work with Langston such as [FL89]. I remembered that we tried to prove that graphs of bounded treewidth were well quasi-ordered by the minor ordering using methods from automata theory. This is a cool project that has never worked out, but is still a fascinating possibility. We also worked on expanding our repertoire of hardness results into other combinatorial problems. In particular, at that time we worked on applying this to Angluin-type learning complexity with Mike’s student Patricia Evans [DEF93]. I recall meeting Neil Koblitz with whom Mike had been working on parameterized versions of cryptography [FK93]. I think it was in a car trip to Sombrio with Neil in the car that we realized that all problems are kernelizable iff FPT, whereas we had been trying to show that there were some problems in FPT which were not kernelizable. Thinking about this led to the paper with Liming Cai and Jianer Chen (who I had not met at the time) on advice classes of parameterized complexity. (That is, FPT=Polynomial time with slice-wise advice.) This eventually appeared in [CCDF97] due to the enormous backlog in the journal. The recent *WorKer*¹⁸ Leiden talk by Dániel Marx has a lot to say on the issue of Kernelizable=FPT vs e.g. search trees. Dániel has expanded this talk into a contribution to the present volume (Marx [Ma12]).

Of course at the time we had several problems which are still with us. To wit, collapse propagation of the W -hierarchy, approximation (e.g. $2k$) FPT approximation of DOMINATING SET¹⁹, how to deal with space.

¹⁸ A very nice workshop at the Leiden conference center based on topics around the theory of kernelization.

<http://www.lorentzcenter.nl/lc/web/2010/418/info.php3?wsid=418>

¹⁹ To wit, is there an FPT algorithm which either says no size k dominating set or gives a size $2k$ dominating set.

I think at that time Mike's interest had moved into computational biology and string matching, such as LCS. He had the very clever student Mike Hallett, and had connections with Tandy Warnow. I know he was also working with Hans Bodlaender. (Certainly he and I later talked on pattern matching and the material which resulted in [BDFW95] and [BDFHW95].)

During that visit, I began writing [DF93] and did not *Latex* at the time. Mike gave me his source for [DF92a], and said "just do this". Little did I know that I was about to learn *Latex* from someone who was learning disabled in the area. It took me a few year to discover the command "`\begin{theorem}`" or that *Latex* would automatically number things (like theorems, bibitems, etc) or even that there were things called bibitem, cite or ref.

We talked a lot about the programme of getting the work to penetrate. We decided that *if the work was of value then it would be seen to be so at some stage, history would be the judge*. There was no reason that some person working on their own stuff should want to change, *but* if we had a source for the material (like a book) then graduate students would maybe pick it up. After all, writing a book would "surely take only a year or two to write!" Little did we know how wrong that timeframe was to be. I have an e-mail from July 14, 1992 indicating that I was writing a chapter on the Abrahamson-Fellows [AF93] methods, and asking if Mike knew of any FPT applications of the minimization of a submodular function, or of combinatorial optimization. I also mention the beginning of Appendix 1. Clearly the book must have been started by then. So I guess it only took seven years. I also note that around that time Liming Cai and Jianer Chen were definitely in the frame. Almost certainly Mike had visited Jainer, and Liming was a student at the time. They had told us of the Papadimitriou and Yannakakis [PY91] work on MAX SNP and their work showing how it relates to FPT. By the 16th July we were reading Kintala-Fischer and could see how many convolutions are needed if you try to address limited nondeterminism nonparametrically. At the time we were also wondering is there was a decent stratification of the classes using FPT reductions polynomial in both the parameter and the input. It is nice to see this concept resurface in Dániel Marx's material at *WorKer*' 10 (and present in Marx [Ma12]) on bounded search trees where they are called "Polynomial parameter transformations".

Sometime around then, we heard that our paper was accepted by *Structures in Complexity*. We also decided to write up the two papers from that abstract [DF95a, DF95b]. I think Mike wrote the first one and I did the second one.

When I got to Cornell I spoke at the meeting for Anil Nerode's 60th Birthday in maybe May or June. My old advisor John Crossley, co-author Jeff Rempel, and Anil Nerode were in the audience and immediately said what exciting stuff this was. It could be "really important." Nerode said he would support the project in any way he could, especially as an editor. This certainly made Mike (when I told him later) and I very happy as Anil has excellent taste in mathematics and computer science. One of the first papers to appear [DF95c] appears in the volume coming from this meeting.

Both Mike and I were talking about the material all over the place. I spoke in Cornell, Chicago, Urbana, Maryland, George Washington, and a number of other places. Were we pushing Robertson-Seymour too much? Bill Gasarch told me that the W -hierarchy was viewed a bit as the “wierdness hierarchy.”

It is of course natural to view problem classification as a function of logical depth. Perhaps it is the fact that NP is a syntactic class whereas ours were closure under FPT reductions, making them more semantic. There is no difference with respect to the intractability issue. (i.e. if, for instance, $P \neq NP$ then co-NP is not in P so in terms of hardness co-NP hardness shows things hard just as well as NP hardness.) But the difference does surface if you try to, for example, prove an analog of Toda’s Theorem, where there is real trouble with “ $BP \cdot W[P]$ ”. This remains a great set of problems. How do you do randomized computation with only, for example, $k \log n$ bits of nondeterminism? It is also possible to show something like $FPT^{\#W[P]} \supseteq \cup_k AW_k[W[P]]$ where this is the k -th level of the AW-hierarchy, or just $FPT^{\#W[P]} \supseteq AW[*]$. To do this without routing through an analog of BP would entail a new technique to prove $P^{\#P} \supseteq PH$. (That is, proving it or $P^{\oplus P} \supseteq PH$, by counting, but using no probabilistic amplification.) The one parameterized version of Valiant-Vazirani [VV86] by Downey, Fellows and Regan [DFR98] hides things in the reductions. Recent papers by Moritz Müller [Mu08a, Mu08b] describes the issues and are the state of the art.

6 Mr. Feasible

What is the value of Cook’s Theorem? (Or perhaps the “Cook-Levin Theorem”) Why is it significant? The proof that predicate logic is undecidable by using predicates to emulate Turing machines had been around since the 1930’s. By Herbrand’s Theorem, we know that quantifiers can be emulated by infinitary propositional formulas, so in some sense it is hardly surprising that a miniaturization of these ideas can show that CNF SAT is NP-complete. The proof of Cook’s Theorem is hardly difficult, *but is an seminal result*.

The crucial value of Cook’s Theorem is that it tells us *why* things like SAT are hard, assuming NTM ACCEPTANCE is hard. Karp then shows us that hardness is everywhere amongst natural practical problems. In terms of *practical* computing, these two papers must also tell us that, if we are confronted with an NP-complete problem, and we *actually need to solve it*, we should seek other coping strategies. One of the problems with NP-completeness theory is that it really only tells us what *won’t* work, not *how* to tackle the problem.

On the 25th February, 1991, Mike said “As for practical, I don’t know. It’s a bad new theory. Apart from completeness there are some fun positive results...” So whilst we discussed the practicality of the material, at the beginning we did not see the utility of thinking parametrically for *practical computation*²⁰.

²⁰ Even as late as Downey, Fellows and Stege [DFS99], we were saying “the extent to which FPT is really useful us unclear.” We now know that it really is. This can clearly be seen in, for instance, the article by Langston et. al. [LPSSV08].

I cannot recall when we began to realize that FPT gave more than NP. At some stage, we realized that it could be used as a *systematic method of attacking intractability* as later articulated in the paper with Mike’s student Ulrike Stege [DFS98, DFS99]. I believe that it was then that we moved away from Robertson-Seymour and began focusing on kernelization, bounded search trees and the like. Of course, Mike had known of a number of examples before this. (Witness [DF92a].) *But* it was sometime during 1992 that we had kind of an epiphany in this direction.

Personally I believe that this is what is really cool about the area, and why it has flourished. The emphasis went directly towards applications²¹. Our statement of purpose and the delineation of these techniques is clearly found in the paper [DF95c], which we called “Mr Feasible”. We referred to [DFS98] and [DFS99] as “Sons of Feasible.”

When we write the revised teaching version ([DFta]) of the book [DF99], the emphasis will be towards practical considerations. Industrial strength FPT is where we should look. This was in the “manifesto” we called the story of Dr O, in the beginning of the book.

7 1993 and Beyond

In 1993 we continued to develop both the theory and some co-authors. Papadimitriou and Yannakakis [PY93] wrote their paper on VC-dimension and $NP[\log n]$ which is NP with logarithmically many bits of nondeterminism. Later we showed how this fits into our setting. In that paper they mention our work, as they later do in [PY97] as a basis for complexity in database theory, where of course parameterized complexity is a totally natural method of analysis. The real story here was later clarified by Grohe and others as, for example, [Gr01a, Gr01b, Gr02]²²

One key step that remained was to show that the k -STEP HALTING PROBLEM was $W[1]$ -complete. We finally figured the $W[1]$ -completeness in the paper [CCDF96] with Liming and Jianer, I presented to the Sacks Conference in MIT in 1993. Whilst circuits were a nice basis and were likely a good enough basis for an intractability theory, the k -step halting problem was so traditional that it completed the picture of $W[1]$ as a gold standard for hardness. It is unfortunate that we don’t know the situation for $M[1]$ and the acceptance problem for Turing machines of size $k \log n$.

I think the final really fundamental paper was [ADF95], which began with trying to populate $W[P]$, and also attempted a decent parameterized treatment

²¹ Mike Langston is an amazing study in this area. From Robertson-Seymour outer space algorithms to concrete biological computations on supercomputers!

²² I remember being contacted by Martin by e-mail in the mid-90’s. He had worked in bounded-variable logics and canonization. Perhaps because of this he became interested in parameterized complexity. The original investigation of bounded variable logics was in Vardi [Va95]. That paper shows that if you bound the number of queries you get tractability.

of space. We also included some connections with subexponential time, anticipating later developments by Impagliazzo, Paturi and Zane [IPZ01] by half a decade.

Immediately when you consider space, you find that the nondeterministic guessing in the proof that QBFSAT is *PSPACE* complete is far from parametric. What to do? The idea was to use QBFSAT parameterized as the basis for those combinatorial problems usually found to be *PSPACE*-complete. I am not sure that we have a nice treatment of space yet.

Other early authors who picked up the methodology were H. Kaplan, R. Shamir and R. E. Tarjan, [KST94] P. Goldberg, M. Golumbic, H. Kaplan, and R. Shamir [GGKS95], Leizhen Cai [LeC96], and the nice thesis of Bazgan [Baz95], who was the first to connect FPT with approximation.

Later the work penetrated India through Venkatesh Raman, who had independently constructed a parametric reduction for a version of DOMINATING SET and then stumbled on to our papers. He e-mailed Mike and was enthused by Mike's response. Also in the mid-90's, Phokion Kolaitis suggested to Martin Grohe to have a look at our papers.

There were all of Mike's students who were around at the time, and working on aspects and applications of parameterized complexity. Mike Hallett, Mike Dinneen, Ulrike Stege, Patricia Evans, Todd Wareham, Marco Cesati and others.

I am not completely sure why the work did not penetrate as quickly as it might have. The charitable view is that the basic framework was already there, and many of the basic questions were solved, at least those that were accessible. I recall when I first heard of NP-completeness from John Stillwell at Monash University in the late 1970's it was felt that there were "too many" NP-completeness proofs. Perhaps the same held for $W[1]$ and FPT, when the real initiative in the early years was finding all kinds of novel applications of the framework in areas like biology, VLSI, linguistics, pattern matching, robot motion planning and the like. At the same time, the small but growing community began to see the simple practical FPT methods being easily codable and widely applicable. *Mr Feasible* was really important. Perhaps this is why the *applications people* took up the ideas.

Unfortunately, Mike and I think there were some casualties because of the lack of penetration into the computational complexity community. This meant that work in the area could be difficult to publish. We believe that a few young people working in the area in its infancy had trouble getting positions. I know that some well-respected authors would not put "parameterized complexity" in the title of their papers if they wanted them to be published in certain conferences/journals *even though what they were working on was parameterized complexity*. Fortunately this situation has all changed in the last decade.

I visited Mike a couple of more times and he visited me in Wellington two more times also, once with Ulrike. We worked on the material on coding theory (appearing later as [DFVW99],) the structural question about the W^* -hierarchy

([DFT96, DF98]) where the parameter gives the depth of the circuits as part of the input²³, the mission statement [DFS98], and of course the book [DF99].

8 Epilogue

By about 1994-1995, the basic papers were done and we had clear paths to develop. The insight that we could have this extended conversation with a problem parametrically, and the development of the distinctive tools such as those in Mr Feasible and his son, but also later ones like Colour Coding, crown reductions, iterative compression have all enriched the subject. We planned to include all the basic material around at the time on implicit parameters like treewidth, and particularly the work of Bodlaender and his co-authors, such as Bodlaender [Bod93, Bod96].

By the time the book [DF99] was published, I felt somewhat bruised. I was ready for an affair with another siren to obsess over and fell into working in Algorithmic Randomness culminating in yet another book (after having vowed never to write one again) [DH10]²⁴. Actually, Kolmogorov complexity is a reasonable parameter to look at in graph theory. It would seem that graphs of high Kolmogorov complexity ought to have some kind of 0-1 law for their algorithmic behaviour. This has not been explored.

Whilst I have kept a running interest in our child's development, serving on PC's, reading new papers and the most excellent books of Niedermeier [Nie06], and Flum and Grohe [FG06], writing one or two papers a year, particularly with new developments like online parameterization, parameterized inapproximability, $M[1]$ and the kernel lower bound project, Mike has really been the beating heart of the subject, spreading the word.

For me the speed and depth of the mathematical developments of the 2010 *WorKer* meeting in Leiden was kind of scary. I had been finishing the randomness book, and looked away from the subject for a year, and arrived to find that some amazingly dramatic progress had occurred. Mainly through the activities of a number of very talented young people. Maybe I am just getting old.

On the other hand, as a community maybe we should take Mr Feasible as a lesson, and not get too obsessed with the beauty of complicated mathematics, so as to lose sight of the *practicality* of what we are trying to do.

It was incredibly fun and rewarding to be involved in all those papers long ago. I like to think the idea of parameterized complexity will be of lasting value to practical computation. Looking at all the young and clever people at the Leiden conference I think the subject is now in excellent hands.

And to finish "Congratulations Mike!"

²³ Notably it is still open if $W^*[t] = W[t]$ for $t \geq 3$.

²⁴ By a strange twist of fate, the motivating question for my working in algorithmic randomness was a Ladner-style density question for another degree structure about the "degree of randomness" called Solovay reducibility, as I articulated in the preface to [DH10].

References

- [ADF93] Abrahamson, K., Downey, R., Fellows, M.: Fixed Parameter Intractability II (Extended Abstract). In: Enjalbert, P., Wagner, K.W., Finkel, A. (eds.) STACS 1993. LNCS, vol. 665, pp. 374–385. Springer, Heidelberg (1993)
- [ADF95] Abrahamson, K., Downey, R., Fellows, M.: Fixed Parameter Tractability and Completeness IV: On Completeness for $W[P]$ and $PSPACE$ Analogs. *Annals of Pure and Applied Logic* 73, 235–276 (1995)
- [AEFM89] Abrahamson, K., Ellis, J., Fellows, M., Mata, M.: On the complexity of fixed-parameter problems. In: *Proceedings of 13th FOCS*, pp. 210–215 (1989)
- [AF93] Abrahamson, K., Fellows, M.: Finite Automata, Bounded Treewidth and Wellquasiordering. In: *Graph Structure Theory. Contemporary Mathematics Series*, vol. 147, pp. 539–564. American Mathematical Society (1993)
- [AR01] Alekhovich, M., Razborov, A.: Resolution is Not Automatizable Unless $W[P]$ is Tractable. In: *Proc. of the 42nd IEEE FOCS*, pp. 210–219 (2001)
- [AYZ94] Alon, N., Yuster, R., Zwick, U.: Color-Coding: A New Method for Finding Simple Paths, Cycles and Other Small Subgraphs Within Large Graphs. In: *Proc. Symp. Theory of Computing (STOC)*, pp. 326–335. ACM (1994)
- [BDG87] Balcazar, J., Diaz, J., Gabarro, J.: *Structural Complexity*, vol. 1. Springer (1987)
- [Baz95] Bazgan, C.: Schémas d’approximation et complexité paramétrée. Rapport de stage de DEA d’Informatique à Orsay (1995)
- [Bod93] Bodlaender, H.L.: A linear time algorithm for finding tree-decompositions of small treewidth. In: *Proceedings of the 25th ACM Symposium on Theory of Computing*, pp. 226–234 (1993)
- [Bod96] Bodlaender, H.L.: A linear time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal on Computing* 25, 1305–1317 (1996)
- [BDFH08] Bodlaender, H., Downey, R., Fellows, M., Hermelin, D.: On Problems without Polynomial Kernels (Extended Abstract). In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfssdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part I. LNCS, vol. 5125, pp. 563–574. Springer, Heidelberg (2008); Final version to appear in *Journal of Computing and System Sciences*
- [BDFHW95] Bodlaender, H., Downey, R., Fellows, M., Hallett, M., Wareham, H.T.: Parameterized Complexity Analysis in Computational Biology. *Computer Applications in the Biosciences* 11, 49–57 (1995)
- [BDFW95] Bodlaender, H., Downey, R., Fellows, M., Wareham, H.T.: The Parameterized Complexity of the Longest Common Subsequence Problem. *Theoretical Computer Science A* 147, 31–54 (1995)
- [BF95] Bodlaender, H., Fellows, M.: On the Complexity of k -Processor Scheduling. *Operations Research Letters* 18, 93–98 (1995)
- [BFH94] Bodlaender, H., Fellows, M.R., Hallett, M.T.: Beyond NP-completeness for Problems of Bounded Width: Hardness for the W Hierarchy. In: *Proc. ACM Symp. on Theory of Computing (STOC)*, pp. 449–458 (1994)
- [La75] Ladner, R.: On the Structure of Polynomial Time Reducibility. *Journal of the Association for Computing Machinery* 22, 155–171 (1975)
- [LeC96] Cai, L.: Fixed-parameter tractability of graph modification problems for hereditary properties. *Information Processing Letters* 58(4), 171–176 (1996)
- [CC97] Cai, L., Chen, J.: On Fixed-Parameter Tractability and Approximability of NP-Hard Optimization Problems. *J. Computer and Systems Sciences* 54, 465–474 (1997)

- [CCDF96] Cai, L., Chen, J., Downey, R.G., Fellows, M.R.: On the Parameterized Complexity of Short Computation and Factorization. *Arch. for Math. Logic* 36, 321–337 (1997)
- [CCDF97] Cai, L., Chen, J., Downey, R., Fellows, M.: Advice Classes of Parameterized Tractability. *Annals of Pure and Applied Logic* 84, 119–138 (1997)
- [Co87] Courcelle, B.: Recognizability and Second-Order Definability for Sets of Finite Graphs. Technical Report I-8634, Universite de Bordeaux (1987)
- [CDF97] Courcelle, B., Downey, R., Fellows, M.: A Note on the Computability of Graph Minor Obstruction Sets for Monadic Second Order Ideals. *Journal of Universal Computer Science* 3, 1194–1198 (1997)
- [CW95] Cesati, M., Wareham, H.T.: Parameterized Complexity Analysis in Robot Motion Planning. In: *Proceedings 25th IEEE Intl. Conf. on Systems, Man and Cybernetics*, vol. 1, pp. 880–885. IEEE Press, Los Alamitos (1995)
- [Do12] Downey, R.: A Basic Parameterized Complexity Primer. In: Bodlaender, H.L., et al. (eds.) *Fellows Festschrift. LNCS*, vol. 7370, pp. 91–128. Springer, Heidelberg (2012)
- [DEF93] Downey, R., Evans, P., Fellows, M.: Parameterized Learning Complexity. In: *Proc. 6th ACM Workshop on Computational Learning Theory*, pp. 51–57 (1993)
- [DF91] Downey, R., Fellows, M.: A completeness theory for fixed parameter problems, March 25 (1991) (unpublished manuscript)
- [DF92a] Fellows, M.R.: Fixed parameter tractability and completeness. *Congressus Numerantium* 87, 161–187 (1992)
- [DF92b] Downey, R., Fellows, M.: Fixed parameter intractability. In: *Proceedings Structure in Complexity, Seventh Annual Conference*, pp. 36–50. IEEE Publication (1992)
- [DF93] Downey, R., Fellows, M.: Fixed Parameter Tractability and Completeness III: Some Structural Aspects of the W -Hierarchy. In: Ambos-Spies, K., Homer, S., Schöning, U. (eds.) *Complexity Theory: Current Research*, pp. 166–191. Cambridge Univ. Press (1993)
- [DF95a] Downey, R.G., Fellows, M.R.: Fixed Parameter Tractability and Completeness I: Basic Theory. *SIAM Journal of Computing* 24, 873–921 (1995)
- [DF95b] Downey, R.G., Fellows, M.R.: Fixed Parameter Tractability and Completeness II: Completeness for $W[1]$. *Theoretical Computer Science A* 141, 109–131 (1995)
- [DF95c] Downey, R.G., Fellows, M.R.: Parametrized Computational Feasibility. In: Clote, P., Remmel, J. (eds.) *Feasible Mathematics II*, pp. 219–244. Birkhauser, Boston (1995)
- [DF98] Downey, R.G., Fellows, M.: Threshold Dominating Sets and an Improved Characterization of $W[2]$. *Theoretical Computer Science* 209, 123–140 (1998)
- [DF99] Downey, R.G., Fellows, M.R.: *Parameterized Complexity*. Springer (1999)
- [DFta] Downey, R.G., Fellows, M.R.: *Fundamentals of Parameterized Complexity*. Springer (in preparation, 2012)
- [DFS98] Downey, R., Fellows, M., Stege, U.: Parameterized Complexity: A Framework for Systematically Confronting Computational Intractability. In: Graham, R., Krachovil, J., Nesetril, J., Roberts, F. (eds.) *Contemporary Trends in Discrete Mathematics. DIMACS*, vol. 49, pp. 49–100. American Mathematical Society (1999)
- [DFS99] Downey, R., Fellows, M., Stege, U.: Computational Tractability: the View from Mars. *Bulletin of the European Association for Theoretical Computer Science* (69), 73–97 (1999)

- [DFKHW94] Downey, R.G., Fellows, M., Kapron, B., Hallett, M., Wareham, H.T.: The Parameterized Complexity of Some Problems in Logic and Linguistics. In: Matiyasevich, Y.V., Nerode, A. (eds.) LFCS 1994. LNCS, vol. 813, pp. 89–100. Springer, Heidelberg (1994)
- [DFR98] Downey, R.G., Fellows, M.R., Regan, K.W.: Parameterized Circuit Complexity and the W Hierarchy. *Theoretical Computer Science A* 191, 91–115 (1998)
- [DFT96] Downey, R.G., Fellows, M., Taylor, U.: The Parameterized Complexity of Relational Database Queries and an Improved Characterization of $W[1]$. In: *Combinatorics, Complexity and Logic: Proceedings of DMTCS 1996*, pp. 194–213. Springer (1997)
- [DFVW99] Downey, R., Fellows, M., Vardy, A., Whittle, G.: The Parameterized Complexity of Some Fundamental Problems in Coding Theory. *SIAM J. Comput.* 29, 545–570 (1999)
- [DFo03] Downey, R., Fortnow, L.: Uniformly hard languages. *Theoretical Computer Science* 298(2), 303–315 (2003)
- [DGHM89] Downey, R., Gasarch, W., Homer, S., Moses, M.: Honest polynomial reductions, non-relativizations and $P = ?NP$. In: *Proceedings of the 4th Annual Conference on Structures in Complexity Theory*, pp. 196–207. IEEE Publ. (1989)
- [DH10] Downey, R., Hirschfeldt, D.: *Algorithmic Randomness and Complexity*, pp. xvii+855. Springer (2010)
- [DM08] Downey, R., Montalbán, A.: The isomorphism problem for torsion-free abelian groups is analytic complete. *Journal of Algebra* 320, 2291–2300 (2008)
- [Fe89] Fellows, M.R.: The Robertson-Seymour Theorems: a Survey of Applications. In: *Contemporary Mathematics*, vol. 89, pp. 1–18. AMS (1989)
- [FL87] Fellows, M.R., Langston, M.: Nonconstructive Proofs of Polynomial-Time Complexity. *Information Processing Letters* 26(88), 157–162 (1987/1988)
- [FL88] Fellows, M.R., Langston, M.: Nonconstructive Tools for Proving Polynomial-Time Complexity. *Journal of the Association for Computing Machinery* 35, 727–739 (1988)
- [FL89] Fellows, M.R., Langston, M.A.: An Analogue of the Myhill-Nerode Theorem and its Use in Computing Finite-Basis Characterizations. In: *Proceedings of the IEEE Symposium on the Foundations of Computer Science*, pp. 520–525 (1989)
- [FL89b] Fellows, M.R., Langston, M.A.: On search, decision and inefficiency of polynomial time algorithms. In: *Proceedings STOC 1989*, pp. 501–512 (1989)
- [FK93] Fellows, M.R., Koblitz, N.: Fixed-Parameter Complexity and Cryptography. In: Moreno, O., Cohen, G., Mora, T. (eds.) *AAECC 1993*. LNCS, vol. 673, pp. 121–131. Springer, Heidelberg (1993)
- [FG06] Flum, J., Grohe, M.: *Parameterized Complexity Theory*. Springer (2006)
- [GJ79] Garey, M., Johnson, D.: *Computers and Intractability: A Guide to the Theory of NP-completeness*. W.H. Freeman, San Francisco (1979)
- [GGKS95] Goldberg, P., Golumbic, M., Kaplan, H., Shamir, R.: Four Strikes Against DNA Physical mapping. *Journal of Computational Biology* 2(1), 139–152 (1995)
- [Gr01a] Grohe, M.: Generalized Model-Checking Problems for First-Order Logic. In: Ferreira, A., Reichel, H. (eds.) *STACS 2001*. LNCS, vol. 2010, pp. 12–26. Springer, Heidelberg (2001)
- [Gr01b] Grohe, M.: The Parameterized Complexity of Database Queries. In: *Proc. PODS 2001*, pp. 82–92. ACM Press (2001)
- [Gr02] Grohe, M.: Parameterized Complexity for the Database Theorist. *SIGMOD Record* 31(4) (2002)
- [IPZ01] Impagliazzo, R., Paturi, R., Zane, F.: Which problems have strongly exponential complexity? *JCSS* 63(4), 512–530 (2001)

- [KW90] Kannan, S., Warnow, T.: Inferring Evolutionary History from DNA Sequences. In: Proceedings of the 31st Annual Symposium on the Theory of Computing, pp. 362–378 (1990)
- [KST94] Kaplan, H., Shamir, R., Tarjan, R.E.: Tractability of Parameterized Completion Problems on Chordal and Interval Graphs: Minimum Fill-In and DNA Physical Mapping. In: Proc. 35th Annual Symposium on the Foundations of Computer Science (FOCS), pp. 780–791. IEEE Press (1994)
- [KF80] Kintala, C., Fischer, P.: Refining nondeterminism and relativized polynomial time bounded computations. *SIAM J. Comput.* 9, 46–53 (1980)
- [KR00] Khot, S., Raman, V.: Parameterized Complexity of Finding Subgraphs with Hereditary properties. *Theoretical Computer Science* 289, 997–1008 (2002); preliminary version in: Du, D.-Z., Eades, P., Sharma, A.K., Lin, X., Estivill-Castro, V. (eds.) COCOON 2000. LNCS, vol. 1858, pp. 137–147. Springer, Heidelberg (2000)
- [LPSSV08] Langston, M., Perkins, A., Saxton, A., Scharff, J., Voy, B.: Innovative computational methods for transcriptomic data analysis: A case study in the use of FPT for practical algorithm design and implementation. *The Computer Journal* 51(1), 26–38 (2008)
- [Le83] Lenstra, H.: Integer Programming with a Fixed Number of Variables. *Mathematics of Operations Research* 8, 538–548 (1983)
- [LP85] Lichtenstein, O., Pnueli, A.: Checking that Finite State Concurrent Programs Satisfy their Linear Specification. In: POPL 1985, pp. 97–107 (1985)
- [MR99] Mahajan, M., Raman, V.: Parameterizing Above Guaranteed Values: MaxSat and MaxCut. *J. Algorithms* 31, 335–354 (1999)
- [Ma12] Marx, D.: What’s Next? Future Directions in Parameterized Complexity. In: Bodlaender, H.L., Downey, R., Fomin, F.V., Marx, D. (eds.) *Fellows Festschrift*. LNCS, vol. 7370, pp. 469–496. Springer, Heidelberg (2012)
- [Mo99] Mohar, B.: A Linear Time Algorithm for Embedding Graphs in an Arbitrary Surface. *SIAM J. Discrete Math.* 12, 6–26 (1999)
- [Mu08a] Müller, M.: Parameterized Derandomization. In: Grohe, M., Niedermeier, R. (eds.) *IWPEC 2008*. LNCS, vol. 5018, pp. 148–159. Springer, Heidelberg (2008)
- [Mu08b] Müller, M.: Valiant-vazirani lemmata for various logics. *Electronic Colloquium on Computational Complexity (ECCC)* 15(063) (2008)
- [Nie06] Niedermeier, R.: *Invitation to Fixed-Parameter Algorithms*. Oxford University Press (2006)
- [OW91] Ogiwara, M., Watanabe, O.: On Polynomial Bounded Truth-Table Reducibility of NP Sets to Sparse Sets. *SICOMP*, 471–483 (1991)
- [PY91] Papadimitriou, C., Yannakakis, M.: Optimization, approximation, and complexity classes. *J. Comput. Syst. Sci.* 43, 425–440 (1991)
- [PY93] Papadimitriou, C., Yannakakis, M.: On Limited Nondeterminism and the Complexity of the V-C Dimension. In: *Eight Annual Conference on Structure in Complexity Theory*, pp. 12–18 (1993)
- [PY97] Papadimitriou, C., Yannakakis, M.: On the Complexity of Database Queries. In: *Proc. ACM Symp. on Principles of Database Systems*, pp. 12–19 (1997); Journal version in *Journal of Computer System Sciences* 58, 407–427 (1999)
- [ST98] Shamir, R., Tzur, D.: The Maximum Subforest Problem: Approximation and Exact Algorithms. In: *Proc. ACM Symposium on Discrete Algorithms, SODA 1998*, pp. 394–399. ACM Press (1998)
- [St00] Stege, U.: *Resolving Conflicts in Problems in Computational Biochemistry*. Ph.D. dissertation, ETH (2000)

- [Ra97] Raman, V.: Parameterized Complexity. In: Proceedings of the 7th National Seminar on Theoretical Computer Science, Chennai, India, pp. 1–18 (1997)
- [Re89] Regan, K.: Finite substructure languages. In: Proceedings 4th Structure in Complexity Annual Conference, pp. 87–96 (1989)
- [RS86a] Robertson, N., Seymour, P.D.: Graph minors. II. Algorithmic aspects of tree-width. *Journal of Algorithms* 7, 309–322 (1986)
- [VV86] Valiant, L., Vazirani, V.: NP is as easy as detecting unique solutions. *Theoret. Comput. Sci.* 47, 85–93 (1986)
- [Va82] Vardi, M.: The Complexity of Relational Database Queries. In: Proc. STOC, pp. 137–146 (1982)
- [VW86] Vardi, M., Wolper, P.: An Automata-Theoretic Approach to Automatic Program Verification. In: LICS 1986, pp. 332–344 (1986)
- [Va95] Vardi, M.: On the complexity of bounded-variable queries. In: PODS 1995 (1995)
- [Va09] Vardi, M.: Conferences vs. Journals in Computing Research. *Communications of the ACM* 52(5), 5 (2009)
- [Ya95] Yannakakis, M.: Perspectives in Database Theory. In: FOCS, pp. 224–246 (1995)

Crypto Galore!

Neal Koblitz

Department of Mathematics, Box 354350
University of Washington
Seattle, WA 98195 U.S.A.
koblitz@math.washington.edu

Abstract. I discuss three aspects of mathematical cryptography that have been themes of Mike Fellows' work: applications of parameterized complexity, combinatorial systems, and Kid Krypto. At times my treatment is anecdotal, and on occasion it veers toward the impractical, fanciful, and even downright goofy.

*Dedicated to Mike Fellows
on the occasion of his 60th birthday.*

1 Crypto and Parameterized Complexity

In [12] Mike and I took a first step in applying parameterized complexity theory to cryptography. The general idea was that in cryptography one might want certain parameters to grow very large, while others are either fixed or vary within a small range. Such a situation is not accurately modeled by traditional complexity theory; parameterized complexity, we thought, might be better suited to formalizing the study of this sort of multitiered dependence on parameters.

In [12] we did not set out to obtain new cryptographic results. Rather, we wanted to formalize some results and conjectures that were already well known to cryptographers on an intuitive level. The rigorous language of parameterized complexity theory would, we hoped, help make the study of these questions more precise and systematic.

1.1 Factorization and Discrete Logarithms

Since the invention of public-key cryptography in the 1970s the two most important conjecturally-hard computational problems upon which public-key protocols have been based have been:

- Integer Factorization, which in the setting of RSA [28] means the problem of finding the factorization of a large integer N that is the product of two secret primes p and q .
- Discrete Logarithm (see [9]), which in the setting of a subgroup \mathbb{G} of prime order q of the multiplicative group of a prime field \mathbb{F}_p means the problem, given $g, y \in \mathbb{G}$, of finding an integer x such that $y = g^x$.

Even though at first glance these two problems seem to be totally unrelated to each other, in the 1980s it turned out (see [32]) that advances in solving integer factorization almost always led to similar advances — and similar running times — for discrete log algorithms in \mathbb{F}_p .¹

There was, however, one important exception. In 1985 Hendrik Lenstra announced his elliptic curve method for integer factorization (see [24]), which was a strikingly new and elegant approach to factoring. Its appeal was not that it could factor an RSA modulus N much faster than the best competing algorithm available at the time (the quadratic sieve) — it could not. Rather, its appeal was two-fold: (1) for the first time, sophisticated mathematics was used in an important way in cryptography, and (2) Lenstra’s algorithm was the only subexponential one whose running time depends not so much on the bitlength of N as on the bitlength of its smallest prime divisor p ; in fact, its heuristic running time is $(\log^2 N) \exp\left(\sqrt{(2 + o(1)) \log p \log \log p}\right)$.

In [12] we considered the following parameterized search problem (with parameter denoted k).

Bounded Factor Factorization.

Input: A parameter k and an integer N that has a prime factor $p < n^k$, where n is the bitlength of N .

Output: A prime factor $p < n^k$.

By making a modification in Lenstra’s algorithm, we showed that under a very plausible number-theoretic conjecture (about smooth integers in short intervals) the problem Bounded Factor Factorization is randomized fixed-parameter tractable. In contrast, in relation to Discrete Logarithm we looked at two open questions where one might be tempted to conjecture fixed-parameter intractability.

Bounded Size Discrete Logarithm.

Input: A parameter k , an n -bit prime p , a subgroup $\mathbb{G} \subset \mathbb{F}_p^\times$ whose order is a prime q , an element $g \in \mathbb{G}$, and an element $y \in \mathbb{G}$.

Output: The answer to the question: Is there a positive integer $x < n^k$ such that $y = g^x$?

Bounded Hamming Weight Discrete Logarithm.

Input: Same as in Bounded Size Discrete Logarithm.

Output: The answer to the question: Is there a positive integer x whose binary representation has at most k 1’s such that $y = g^x$?

We noted that in both cases tractability of the decision problem is equivalent to that of the corresponding search problem, thanks to a “self-reducible” property of discrete logs. We also remarked that the Bounded Hamming Weight problem is of practical interest, because in some situations there might be an improvement in efficiency if one uses only exponents of bounded Hamming weight.

¹ In more recent times the use of $\mathbb{G} \subset \mathbb{F}_p^\times$ has been largely abandoned in favor of subgroups of elliptic curve groups. However, those groups are not relevant to this discussion.

But one would not want to do this if it would enable an adversary to find the discrete log much faster.

In [12] we then asked the questions: Is Bounded Size Discrete Logarithm randomized fixed-parameter tractable? How about Bounded Hamming Weight Discrete Log? To the best of our knowledge the answer in both cases is likely to be no — in contrast to Bounded Factor Factorization considered above.

Essentially what we were suggesting was that parameterized complexity might separate the factorization and discrete log problems, whereas conventional complexity theory did not.

Unfortunately, as far as I know the paper [12] didn't lead to anything. I know of no subsequent work that viewed cryptographic complexity issues from the standpoint of parameterized complexity theory. I don't think this is because there is nothing to be gained from such an approach. Rather, the explanation for the lack of interest is probably sociological: the set of cryptographers and the set of experts on parameterized complexity form two disjoint subcultures of computer science.

In the next subsection I'll give an example of an important current concept in cryptography that might benefit from a parameterized-complexity point of view.

1.2 The Bounded Retrieval Model

An information security issue that has become increasingly important in recent years is protection against so-called “side channel” attacks. These are attacks that are based not on weaknesses in the mathematical one-way function or the protocol design, but rather on an analysis of such features of the hardware as electromagnetic radiation emitted, the amount of power or time used, or the effects of induced faults. The problem of side channel vulnerability of cryptographic devices goes back a long way — in fact, to World War II — but theoreticians didn't start to address the issue until just a few years ago. The term “leakage resilience” is now used to refer to the property of remaining secure even if a limited amount of secret data leaks to the adversary. Leakage resilience has become a “hot” area at crypto conferences.

Perhaps the most interesting concept that has been introduced in the leakage resilience literature is that of the “bounded retrieval model,” developed in [7,8,10] (see also [2,3,16]). Elsewhere [23] Menezes and I have raised doubts about whether the bounded retrieval model is really a useful approach to dealing with side channel attacks. However, there is no doubt that it is a clever and nifty idea. It is worthy of study for that reason alone even if it turns out ultimately to be of no practical use whatsoever.

The idea of the bounded retrieval model is simple and elegant. Suppose that an adversary is able somehow to acquire a certain amount of secret key data. One countermeasure is to increase the sizes of all of the parameters of the cryptosystem so that any secret key becomes larger than what the adversary is likely to be able to capture. This would result in extremely inefficient cryptography.

But what if each user Alice could generate and store a very large number M of secret keys, all corresponding to the same public key, and in carrying out a

protocol she would use only one of the those keys or a small randomly chosen subset of them (a clue sent by her correspondent Bob would tell her which of her secret keys she needs to use)? This stable of secret keys could be huge; M is limited only by available storage and the time required to generate all of the M keys during the initial set-up stage. Meanwhile, the size of the public key and everything else (ciphertexts, signatures, etc.) — and the amount of time to carry out protocols — would be essentially fixed and independent of the number of secret keys. This is what the bounded retrieval model requires. Now the adversary’s task becomes much more difficult, because any secret key material he acquired would be unlikely to be sufficient for breaking a given execution of the protocol. However, the cost to Alice is quite reasonable.

I don’t know whether fixed-parameter complexity applies directly to this situation. However, the “philosophy” of fixed-parameter complexity seems to be very relevant. One way to formalize complexity in the bounded retrieval model would be by analogy with the NC classes used to study parallel computation. That is, as a function of the one parameter that becomes enormous — namely, M — the running times of encryption, decryption, signature, etc. must be poly-logarithmic, whereas as a function of the other parameters these running times would not have to be so tightly controlled; they could be polynomial or even superpolynomial. To put it another way, all of the parameters could be regarded as essentially fixed except for the number M of secret keys, and the running time for each execution of the protocol would have to be polynomial in $\log M$.

I cannot say whether or not the bounded retrieval model would be a fruitful area for fixed-parameter complexity research. My purpose in bringing it up is to illustrate how the general notion of fixed-parameter complexity might have natural applications in cryptography.

2 Combinatorial Crypto Galore

2.1 Knapsacks and Brassard’s Theorem

In the late 1970s a combinatorial cryptosystem called the Merkle-Hellman Knapsack [26], based on the Subset Sum problem, was viewed as holding great promise. For public-key encryption it was much more efficient than its main competitor at the time, which was RSA. In addition, it was thought to be almost *provably* secure, since the Subset Sum problem is NP-complete. However, within a few years Shamir [30] completely broke Merkle-Hellman, showing that the subproblem of Subset Sum that its security relies upon can be solved in polynomial time. Although modifications and generalizations were developed in an attempt to salvage the situation, in the 1980s most of them were also broken by Shamir, Brickell, Odlyzko, and others [6,27]. Many cryptographers were traumatized by this experience, which at the very least taught the lesson that NP-complete problems should be used with great caution. Perhaps it would be better, many people thought, to stick with number-theoretic problems such as Integer Factorization and Discrete Logarithm, which are believed not to be NP-hard.

There was a second reason why cryptosystems based on NP-complete combinatorial problems fell into disfavor. A theorem of Brassard [5] published in 1979 said — or, rather, was interpreted as saying — that if breaking a cryptosystem is NP-hard, then $\text{NP}=\text{coNP}$, which nobody believes is true. A widespread consensus was that, in Selman’s words [29], “There can be no hope to transform arbitrary problems in $\text{NP} \setminus \text{P}$ into public-key cryptosystems.”

Mike and I thought that this judgment was premature. In the first place, Brassard’s theorem has a strong hypothesis that people were ignoring — namely, the one-way construction upon which the cryptosystem is based must have image in coNP . It seemed to us that Brassard’s theorem is essentially just a circular tautology whose premise is as strong as its conclusion. This premise holds for RSA, where the image of the map $(p, q) \mapsto N = pq$ from

$$\text{Primes} \times \text{Primes} \longrightarrow \text{Integers}$$

is clearly in coNP . However, the hypothesis of Brassard’s theorem about the image being in coNP seems to be false for most of the combinatorial constructions that have been proposed for public-key cryptography.

In the second place, in [13,14] Mike and I showed how to convert *any* NP search problem into a system for public-key encryption that is potentially hard to break if the problem is NP-hard. We titled the paper [14] “Combinatorial cryptosystems galore!” (“galore” was Mike’s word) as a way of exuberantly rejecting the view expressed by Selman.

A public-key cryptosystem is based on a mathematical problem that is believed to be intractable. However, not just any old difficult mathematical problem will necessarily lead to a cryptosystem. One has to find a way to convert the problem to a “one-way” encryption function $y = f(x)$ that without knowledge of the secret key is easy to compute but infeasible to invert. For example, the underlying hard mathematical problem for RSA is factorization of an integer N that’s the product of two primes, in other words, inversion of the above map $(p, q) \mapsto N = pq$. However, this does not immediately give us a way to encipher messages — it took some cleverness on the part of Rivest, Shamir, and Adleman to come up with an encryption function. The enciphering function $y = f(x)$ in this case is exponentiation modulo N : $y = x^e \pmod{N}$, where e is a publicly known exponent. Inverting this function by finding a decryption exponent d for which $x = y^d \pmod{N}$ is easy if you know the factorization of N (which is the secret key) but is infeasible otherwise.

The method Mike and I used to construct a cryptosystem from an NP problem was basically to algebraicize the problem — something that researchers had been profitably doing in other areas [1,25]. We worked with multivariate polynomial ideals that were constructed in such a way that finding a point where the ideal vanishes is equivalent to solving the NP search problem. Alice uses a one-way construction to come up with a hard instance of a problem for which she knows a solution, i.e., a point where the corresponding ideal vanishes. Then in order to send a message m to Alice, Bob randomly chooses a complicated polynomial in her ideal and then adds m to it. Alice deciphers the message by evaluating the polynomial at her secret point. Rather than giving details in this general

context, let's look at how the cryptosystem works in a particular setting. Let's take the case of graph 3-coloring.

A public-key system based on graph 3-colorability. Let $G = (V, E)$ be a graph with vertex set V and edge set E . Let T be a set of variables with three variables $t_{v,1}, t_{v,2}, t_{v,3}$ corresponding to each vertex $v \in V$. Let I be the ideal of the polynomial ring $\mathbb{Z}[T]$ with the following basis $B = B_1 \cup B_2 \cup B_3$:

$$B_1 = \{t_{v,1} + t_{v,2} + t_{v,3} - 1 : v \in V\};$$

$$B_2 = \{t_{v,i}t_{v,j} : v \in V, 1 \leq i < j \leq 3\};$$

$$B_3 = \{t_{u,1}t_{v,1} + t_{u,2}t_{v,2} + t_{u,3}t_{v,3} : uv \in E\}.$$

The vertex coloring that assigns the color i_v ($1 \leq i_v \leq 3$) to the vertex $v \in V$ is associated with the point whose t_{v,i_v} -coordinate is equal to 1 and whose other two coordinates corresponding to the variables $\{t_{v,1}, t_{v,2}, t_{v,3}\}$ are equal to 0. One easily checks that a point is in the affine variety $V(I)$ (that is, every polynomial in I vanishes at the point) if and only if the vertex coloring is a proper 3-coloring of G .

Alice constructs a public/private key pair for encryption by using a “one-way” construction of a graph G for which she knows a 3-coloring. Given a vertex set V , she can start with a random partition into three colors, and then draw a lot of edges none of which connect two vertices in the same color-set. Hopefully she'll be able to do this in such a way that if she keeps the partition secret no one else can figure out a 3-coloring. Alice's public key is the graph G ; this means that the basis B is also public. Her secret key is the 3-coloring (that is, the partition) she started out with in her construction.

Here is how the probabilistic encryption works. Suppose that Bob wants to send Alice a message $m \in \mathbb{Z}$. Using the basis B , by some randomized process he chooses a complicated polynomial in I and adds the integer m to that polynomial. When Alice receives this ciphertext, all she has to do is evaluate it at the point corresponding to her secret 3-coloring; in other words, she sets $t_{v,i}$ equal to 1 if v is colored i and equal to 0 otherwise. This makes Bob's random polynomial disappear, revealing his message m .

2.2 Polly Cracker

The NP-hard problem where we thought the most about this hybrid algebraic/combinatorial construction was Perfect Code. This is the problem of finding a subset $V' \subset V$ of vertices such that every $v \in V$ is in the neighborhood $N[v']$ of one and only one $v' \in V'$. Perfect Code is NP-hard even when restricted to 3-regular graphs.

Alice can construct an instance of a graph with a perfect code as follows. Given a vertex set V , she first randomly chooses a subset $V' \subset V$ of suitable size. Then she forms “stars” emanating from the vertices in V' such that every $v \in V$ is in one and only one star. Finally, she disguises the locations of the $v' \in V'$ by drawing a lot of additional edges between vertices in $V \setminus V'$. The result is her public key $G = (V, E)$. Her secret key is the subset V' .

The algebraicization of Perfect Code is the ideal in $\mathbb{Z}[T]$, where $T = \{t_v : v \in V\}$, with basis $B = B_1 \cup B_2$, where

$$B_1 = \{1 - \sum_{u \in N[v]} t_u : v \in V\};$$

$$B_2 = \{t_u t_{u'} : u, u' \in N[v], u \neq u', v \in V\}.$$

A large part of the appeal of this cryptosystem was the name that Mike thought of for it: Polly Cracker. I also devoted a chapter of my book [19] to these constructions.

Over the years there have been a lot of cruddy, worthless cryptosystems that have never been broken because no one thought it was worth their time to go to the effort. This was not, however, the fate of Polly Cracker. Soon a number of cryptographers started looking for ways to break Polly Cracker without solving the underlying instance of an NP-hard problem. Hendrik Lenstra was the first to note that it might be possible to recover a message from Bob's ciphertext using "intelligent linear algebra." I tried to suggest some guidelines for Bob's choice of polynomials that might resist such an attack. However, it eventually became clear that such an effort was hopeless. After some articles were published with rather comprehensive attacks (see, for example, [31]), there was no denying that Polly Cracker was cracked.

Note Added in Proof: I was too hasty in saying that the only good thing about the Polly Cracker cryptosystem was its name. Recently versions of Polly Cracker that are apparently both useful and secure have been proposed in two papers by Albrecht *et al.* and by Caboara *et al.*; see <http://eprint.iacr.org/2011/289.pdf> and <http://academic.research.microsoft.com/Paper/5626032>

3 Kid Krypto

Polly Cracker lives on, however, in a simplified form that works well with children. In the version described above suppose that we restrict Bob's ciphertext to linear polynomials. It's not hard to see that the resulting encryption scheme is equivalent to the following:

Perfect Code Cryptography. Alice's public key is a graph $G = (V, E)$ and her secret key is the location of a perfect code $V' \subset V$. To send Alice a message m , Bob randomly assigns integers (positive or negative) to the vertices of G subject only to the condition that their sum is m . Call these the "blue numbers" b_v , $v \in V$. He then associates a "green number" g_v to each vertex, where each green number is the sum of all neighboring blue numbers: $g_v = \sum_{u \in N[v]} b_u$. His ciphertext is the set of green numbers. Alice deciphers the message by summing the green numbers only over perfect code vertices. This works because

$$\sum_{v \in V'} g_v = \sum_{v \in V} b_v = m.$$

In order to present Perfect Codes to children Mike invented a story. The merchants of Tourist Town are getting ready for the tourist season and have to decide at which street corners to put icecream stands. They want to build as few as possible, except that there have to be enough so that a tourist who's standing at a corner with no icecream doesn't have to walk more than one block to find a stand. Find the minimum number of stands, and decide where to put them.²

When visiting a school in Peru, Mike and I noticed that this story didn't seem to make much sense to the kids. In the first place, Peruvian icecream sellers use movable carts, not fixed stands. Moreover, in any country with large unemployment, where many people depend on the "informal economy" for their livelihood, there is always an overabundance of icecream vendors. The children saw no point in trying to minimize the number of icecream sellers. So we changed the story to *Aldea con Pozos* (Village with Wells), where one wants to minimize the number of wells in order to achieve an efficient water supply for a village or *barrio*. In many Third World settings such a story makes more sense than Tourist Town. Based partly on our experiences in different countries with Kid Krypto and other math enrichment topics, Mike, my wife Ann, and I wrote an article [11] for the *AMS Notices* on "Cultural Aspects of Mathematics Education Reform."

When doing Kid Krypto it's important for the kids not only to understand how and why the encryption works, but also to think critically about how good a system it really is. Is it secure? Supposing that Alice does a good job of constructing a graph with a perfect code that no one is able to figure out, is there any other way an adversary could figure out the message m from the intercepted ciphertext $\{g_v\}_{v \in V}$? Sad to say, the answer is: yes, easily. If the graph is 3-regular, then $\sum_{v \in V} g_v = 4 \sum_{v \in V} b_v = 4m$; in the general case just use linear algebra, regarding the b_v as unknowns and solving the system $\sum_{u \in N[v]} b_u = g_v$ of $|V|$ equations in $|V|$ unknowns.

However, young children don't know linear algebra. So they might use Perfect Code crypto securely until one of them, in her zeal to figure out a secret message, essentially rediscovers a version of gaussian elimination. One can argue — and this is a point that Mike has made many times — that at its best science and math education should be similar to the process of creative research. Linear algebra will be much more meaningful to a student who developed parts of it on her own in order to decipher a secret message than it will be to someone who sat through a boring course of lectures in college. Similarly, in [18] I showed how a "baby version" of RSA that can be broken if one knows how to invert modulo N could motivate students to rediscover the Euclidean algorithm on their own.

Cryptography can be an excellent vehicle for teaching basic concepts in mathematics and computer science. And Mike's Perfect Code cryptosystem allows students with no prerequisites to learn about sophisticated notions such as one-way functions and public-key encryption. It works with almost any age group. I've organized Perfect Code crypto sessions in several middle-school classes, and

² This is actually Minimum Dominating Set of Vertices, but if the given graph has a perfect code, then it amounts to the same thing.

one year I did it as part of the University of Washington’s “Math Day” (see [18] for details). I’ve also used it in a seminar I’ve taught for the past three years to entering students at U.W. (see [21]). Finally, I recently suggested to the developers of an NSF-funded after-school program called “Crypto Club”³ that they incorporate some of Mike’s Kid Krypto into their website.

3.1 Crypto 1992 Invited Talk

When I was on the program committee for Crypto ’92, I lobbied to get Mike on the schedule as an invited speaker so that he could talk about Kid Krypto. I was the session chair during his presentation, in which he explained his ideas on math education (see Tim Bell’s article in this volume for more about that) and the role that cryptography could play (see [4,15]). His talk was entertaining and thought-provoking.

At one point Mike observed that what security means is relative to the mathematical knowledge of the users. A system that can be cracked using college-level math (such as linear algebra) might be secure for use among high school students. Even we adults shouldn’t be too smug about our own knowledge, since to someone from a more advanced civilization it might seem that we’re all doing kid crypto.

To illustrate his point, Mike projected onto the screen a front-page picture from the tabloid *Weekly World News* showing an extraterrestrial creature meeting with then-President Clinton. Mike said that this representative from another galaxy had told Clinton that they’d broken RSA three hundred years ago. I blurted out, “What about ECC?⁴ Did the alien tell him anything about breaking elliptic curves?” The audience laughed at my apparent inability to refrain from interrupting the speaker and blurting out that question.

In reality, I had been with Mike when he bought the *Weekly World News*, and we’d discussed how he would use it in his talk and what would be a funny thing for me to interject. My outburst was part of the plan. Mike has a strong sense of education as theater. This came across in his Crypto ’92 invited talk, just as it has in his visits to schools and math-in-the-park activities.

3.2 P=NP?

When explaining the Perfect Code problem to kids, Mike encourages them to think algorithmically. They should try to come up with a step-by-step method that could be used for any Tourist Town that anyone could make up. He tells them that no one knows of a really good method that would work in all cases, and anyone who found one would become rich and famous. The reason (which he does not mention to the kids) is that, since Perfect Code is NP-hard, a kid who found such an algorithm would have proved that P=NP.

³ It’s based at the University of Illinois at Chicago
(see <http://cryptoclub.math.uic.edu>)

⁴ At this time Elliptic Curve Cryptography, of which I was a passionate advocate, was locked in a bitter rivalry with RSA.

Mike is right: such a kid would become a millionaire. S/he would win not only the million-dollar Clay Millenium Prize, but Scott Aaronson's house as well. This is because, when a purported solution to the $P \neq NP$ problem was announced by V. Deolalikar in August 2010, Aaronson expressed his extreme skepticism by promising to add the value of his house — \$200,000 — to the Clay Millenium Prize if the solution was correct.⁵ Undoubtedly, Scott would be equally skeptical — and again willing to bet his house against it — if it were announced that some 5th-grader had proved that $P=NP$.

4 Retrospective

Mike and I met not through mathematics, but through politics — or, more precisely, through feminism. In the late 1980s Mike heard about the Kovalevskaia Fund,⁶ a small non-profit foundation that my wife Ann directs that aims to support and encourage women in science in developing countries, and wrote us expressing his interest. Then in 1989 when Ann drove across the country from Seattle to take a job at a small college in the East, she stopped for a day with Mike's family in Moscow, Idaho (at that time he was at the University of Idaho). I met Mike soon after, and found that we had a lot of interests in common.

Mike was responsible for my growing involvement in K-12 education during the 1990s. He infected me with his enthusiasm about math enrichment presentations in the schools. It was because of him that Ann and I started arranging visits to schools in the different countries we travel to. During the 1990s we gave math enrichment lessons — often using Mike's examples — in Peru, El Salvador, Belize, Cuba, Mexico, Chile, Vietnam, India, Zimbabwe, Malawi, South Africa, Canada, and the U.S. Twice Mike came along with us to Peru and El Salvador.

Mike has also had a less tangible influence on me. He showed me that there doesn't have to always be a sharp line between one's cultural and political outlook — which for both of us came out of the radical movements of the 1960s — and one's mathematical style. As I commented in my book [22],

Mike, like me, is a product of the 1960s... He...had an effect on my style because he showed that even in a serious professional setting there is room for jokes and a camp sensibility.

I'd like to conclude by blaming Mike for some things I've done that have gotten me into hot water — most notably, my article [20]⁷ (see also <http://anotherlook.ca>). I thought that an article in the *AMS Notices* on the relationship between mathematics and cryptography should be entertaining, humorous, and slightly outrageous.

⁵ See <http://www.scottaaronson.com/blog/?p=456>

⁶ Sofia Kovalevskaia (1850-1891) was the first woman in modern times to earn a doctorate in mathematics. She is best known for the Cauchy-Kovalevskaia theorem, which is basic to the theory of partial differential equations, and for her work on abelian integrals. See [17].

⁷ Available at: <http://www.ams.org/notices/2007008>

It was in part Mike's influence that caused me to adopt that style. To my surprise, my innocent efforts provoked a furious reaction in the computer science blogosphere (especially in the blogs of Luca Trevisan at Berkeley and David Eppstein at Irvine) — and no fewer than five angry letters to the *Notices*.⁸ While licking my wounds from their harsh words, at least I could tell myself that it was all Mike's fault.

Mike has had a long-lasting influence on me. My own background in computer science was weak, and yet in 1985 my interests had shifted from pure math to cryptography, in which computer science was an equal partner (or greater-than-equal partner) with mathematics. For me Mike has been a role model of someone who combines mathematical and computer science ways of thinking and moves seamlessly between the two. It's a rare talent to be able to mediate effortlessly between those two cultures.

Acknowledgments. I wish to thank Ann Hibner Koblitz and Alfred Menezes for helpful comments and suggestions.

References

1. Alon, N., Tarsi, M.: Colorings and orientations of graphs. *Combinatorica* 12, 125–134 (1992)
2. Alwen, J., Dodis, Y., Wichs, D.: Leakage-Resilient Public-Key Cryptography in the Bounded-Retrieval Model. In: Halevi, S. (ed.) *CRYPTO 2009*. LNCS, vol. 5677, pp. 36–54. Springer, Heidelberg (2009)
3. Alwen, J., Dodis, Y., Naor, M., Segev, G., Walfish, S., Wichs, D.: Public-Key Encryption in the Bounded-Retrieval Model. In: Gilbert, H. (ed.) *EUROCRYPT 2010*. LNCS, vol. 6110, pp. 113–134. Springer, Heidelberg (2010)
4. Bell, T., Fellows, M.R., Koblitz, N., Powell, M., Thimbleby, H., Witten, I.: Explaining cryptographic systems to the general public. *Computers and Education* 40, 199–215 (2003)
5. Brassard, G.: A note on the complexity of cryptography. *IEEE Trans. Information Theory* 25, 232–233 (1979)
6. Brickell, E.F.: Breaking Iterated Knapsacks. In: Blakely, G.R., Chaum, D. (eds.) *CRYPTO 1984*. LNCS, vol. 196, pp. 342–358. Springer, Heidelberg (1985)
7. Cash, D., Ding, Y.Z., Dodis, Y., Lee, W., Lipton, R.J., Walfish, S.: Intrusion-Resilient Key Exchange in the Bounded Retrieval Model. In: Vadhan, S.P. (ed.) *TCC 2007*. LNCS, vol. 4392, pp. 479–498. Springer, Heidelberg (2007)
8. Di Crescenzo, G., Lipton, R.J., Walfish, S.: Perfectly Secure Password Protocols in the Bounded Retrieval Model. In: Halevi, S., Rabin, T. (eds.) *TCC 2006*. LNCS, vol. 3876, pp. 225–244. Springer, Heidelberg (2006)
9. Diffie, W., Hellman, M.: New directions in cryptography. *IEEE Trans. Information Theory* 22, 644–654 (1976)
10. Dziembowski, S.: Intrusion-Resilience Via the Bounded-Storage Model. In: Halevi, S., Rabin, T. (eds.) *TCC 2006*. LNCS, vol. 3876, pp. 207–224. Springer, Heidelberg (2006)

⁸ See <http://in-theory.blogspot.com/2007/08/swift-boating-of-modern-cryptography.html>; <http://11011110.livejournal.com/114876.html#cutid1>; <http://www.ams.org/notices/200711>; and <http://www.ams.org/notices/200801>

11. Fellows, M.R., Koblitz, A.H., Koblitz, N.: Cultural aspects of math education reform. *Notices of the Amer. Math. Soc.* 41, 5–9 (1994)
12. Fellows, M.R., Koblitz, N.: Fixed-parameter Complexity and Cryptography. In: Moreno, O., Cohen, G., Mora, T. (eds.) *AAECC 1993*. LNCS, vol. 673, pp. 121–131. Springer, Heidelberg (1993)
13. Fellows, M.R., Koblitz, N.: Combinatorially based cryptography for children (and adults). In: *Proc. 24th Southeastern Intern. Conf. Combinatorics, Graph Theory and Computing*, Boca Raton, Florida (February 1993); *Congressus Numerantium* 99, 9–41 (1994)
14. Fellows, M.R., Koblitz, N.: Combinatorial cryptosystems galore! In: *Finite Fields: Theory, Applications, and Algorithms*, Second Intern. Conf. Finite Fields, Las Vegas (August 1993); *Contemporary Math.* 168, 51–61 (1994)
15. Fellows, M.R., Koblitz, N.: Kid Krypto. In: Brickell, E.F. (ed.) *CRYPTO 1992*. LNCS, vol. 740, pp. 371–389. Springer, Heidelberg (1993)
16. Katz, J.: Signature schemes with bounded leakage resilience, <http://eprint.iacr.org/2009/220.pdf>
17. Koblitz, A.H.: *A Convergence of Lives: Sofia Kovalevskaia — Scientist, Writer, Revolutionary*. Birkhäuser (1983)
18. Koblitz, N.: Cryptography as a teaching tool. *Cryptologia* 21, 317–326 (1997)
19. Koblitz, N.: *Algebraic Aspects of Cryptography*. Springer (1998)
20. Koblitz, N.: The uneasy relationship between mathematics and cryptography. *Notices of the Amer. Math. Soc.* 54, 972–979 (2007)
21. Koblitz, N.: Secret codes and online security: A seminar for entering students. *Cryptologia* 34, 145–154 (2010)
22. Koblitz, N.: *Random Curves: Journeys of a Mathematician*. Springer (2007)
23. Koblitz, N., Menezes, A.J.: Another look at security definitions (to appear), <http://anotherlook.ca>
24. Lenstra Jr., H.W.: Factoring integers with elliptic curves. *Annals Math.* 126, 649–673 (1987)
25. Lund, C., Fortnow, L., Karkoff, H., Nisan, N.: Algebraic methods for interactive proof systems. In: *Proc. 31st IEEE Symp. on Foundations of Computer Science*, pp. 2–10 (1990)
26. Merkle, R., Hellman, M.: Hiding information and signatures in trapdoor knapsacks. *IEEE Trans. Information Theory* 24, 525–530 (1978)
27. Odlyzko, A.: The rise and fall of knapsack cryptosystems. In: *Cryptology and Computational Number Theory*, Proc. Symp. Appl. Math., vol. 42, pp. 75–88. Amer. Math. Soc. (1990)
28. Rivest, R., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM* 21, 120–126 (1978)
29. Selman, A.L.: Complexity issues in cryptography. In: *Computational Complexity Theory*, Proc. Symp. Appl. Math., vol. 38, pp. 92–107. Amer. Math. Soc. (1988)
30. Shamir, A.: A polynomial time algorithm for breaking the basic Merkle Hellman cryptosystem. *IEEE Trans. Information Theory* 30, 699–704 (1984)
31. Steinwandt, R., Geiselmann, W., Endsuleit, R.: Attacking a polynomial-based cryptosystem: Polly Cracker. *Intern. J. Information Security* 1, 143–148 (2002)
32. van Oorschot, P.: A comparison of practical public-key cryptosystems based on integer factorization and discrete logarithm. In: Simmons, G. (ed.) *Contemporary Cryptology: The Science of Information Integrity*, pp. 289–322. IEEE Press (1992)

Flyby: Life Before, During, and After Graduate Studies with Mike Fellows

Todd Wareham

Department of Computer Science
Memorial University of Newfoundland
St. John's, NL A1B 3X5 Canada
harold@mun.ca

I am a child of the space age. Growing up in the 1960's, this was perhaps inevitable. Despite the overwhelming focus on the manned missions to the moon, I was always most fascinated with the deep space planetary probes. The multi-decade journeys of Pioneer and Voyager measured out my high school and university undergraduate years and later the Galileo and Huygens missions saw me through graduate school and becoming faculty. I am currently awaiting the arrival of New Horizons at Pluto in 2015, wondering where and what I'll be then.

The common event in each such mission is, after years of traveling alone through space, a planetary encounter. Such encounters are often flybys, brief visits characterized by a few tantalizing (and possibly unrepresentative) impressions which end when, after stealing some of the gravitational energy locked up in the planet, the probe is flung outward in a new direction, changed forever and never to return.

Graduate studies with Mike Fellows was a lot like that.

I first met Mike over the Internet. On finishing my MSc, I attended the IEEE Structure in Complexity Theory conference in Boston in the summer of 1992. Among the talks I made notes to follow up on when I got home was one given by Rod Downey on parameterized complexity. When I got a chance to look at the conference paper, I realized it was the neatest thing I'd read in ages and decided to ask for some of the manuscripts cited therein. As Mike was the Canadian author and I was in Canada, I wrote to him. Just before I sent the message, I added a brief postscript that I might have solutions to some of the open problems mentioned in the Structure paper. Mike wrote back immediately, promising to put the requested manuscripts in the mail, and, with what I came to realize was his typical generosity, offered to fly me out to Victoria to give a talk.

After several months of e-mails back and forth, I went out to meet Mike. At the airport, I saw his characteristic goofy slightly-open-mouthed grin for the first time. The next three days were a whirlwind, the prototypical Mike Fellows Experience. I saw Mike teach, enthralling an undergraduate class. I got a first-hand taste of his intensity when working, when we spent a day together analyzing the complexity of a graph layout problem from computational biology. What I remember most is Mike talking research, babbling with almost insane energy and joy about all sorts of things I didn't understand (though he kindly assumed

that I did), skipping effortlessly from topic to topic, amazingly free and open with ideas and collaborative opportunities.

On the third day, I was exhausted. After I gave a talk on my MSc work, Mike and I had a very frank chat in the campus grad bar. Though he said I wasn't great at mathematics because I didn't have the killer instinct when working on proofs (with which I agreed), he liked my breadth of interests and offered to supervise me in a PhD. I thanked him, said I'd think about it, and we parted. Over the next six months, we worked on several papers together, I thought a long time about it, and finally decided a PhD with Mike just might work out.

When I arrived in Victoria to start my PhD in January 1994, I was pleasantly startled by both the extraordinarily mild (by my standards) winter weather and being in a computer science department that had a large and vibrant theory group. I got to know Mike's other PhD students, Mike Dinneen (MikeD), Mike Hallett (MikeH), and Patricia Evans (who with Fellows (MikeF) were known as Patricia and the Three Mikes). After we agreed that I would not have to change my name to Mike but could (despite the breaking of convention) remain Todd, I settled into what would become my routine for the next several years — courses, marking, research, evenings at the truly excellent on-campus cinema, and, of course, time spent with Mike Fellows.

Much of that time was spent in Mike's office. It was a corner office on the second floor of the Engineering Office Wing, with two walls as windows looking out on the lush West Coast forest that surrounds the UVic campus and the other two walls as long whiteboards with overflowing bookcases beneath. There were relatively neat piles of papers on every available horizontal surface of sufficient size, often capped with Mike's many manuscripts in progress. Boxes enclosing cryptic descriptions of ongoing and future projects clustered on the edges of each whiteboard, framing the overlapping half-erased scrawls in the centers that characterized Mike's thoughts of the previous month or so. Facing Mike's L-shaped desk was a ragged half-circle of well-used and constantly changing office chairs, which were more often than not occupied. Mike's door was almost always open, and anyone could (and often did) come in, mixing with undergraduate and graduate students and Mike's parade of visitors from other universities.

We had more or less weekly meetings one-and-one with Mike, almost always in the morning, to discuss what we were working on, be it thesis project or a related paper. When we had little to show for the last week, Mike eagerly launched into an energetic explanation of whatever he was working on, with invitations to contribute and be part of the fun. These sessions frequently evolved into impromptu group meetings, sucking in whoever was walking by in the corridor outside. Given Mike's enthusiasm, one-on-one meetings could run long and get a bit intense. If this was a possibility, MikeD, MikeF, Patricia and myself had an agreement that, about 30 or so minutes into the meeting, (1) one of us would walk by Mike's office to see how things were going and, if necessary, (2) distract Mike long enough to give the one in the meeting a chance to either gather their

thoughts or escape. This agreement was infrequently invoked, but did highlight one of the unofficial advantages of large theory groups.

I loved watching Mike teach, and took whatever graduate courses that I could from him. When he intimately knew the topic, as in the Computational Complexity course, it was invariably enthralling. As he was often running a bit behind and had not fully prepared his notes, he would spend the first 10 minutes sketching a story point by point in a stream-of-consciousness soliloquy on the left-hand side of the board, and then (with periodic consultations) give several hours of beautifully-constructed and delivered lecture. I still remember his 2 1/2-hour explanation (if not the details) of the complex chain of parameterized reductions underlying the $W[2]$ -hardness of the Dominating Set problem. When he didn't quite know what he was talking about, as in the Computational Biology course, it was just as fascinating — the unexpected ways he would jumble together those concepts he knew well with those that he didn't, if not always viable, was invariably both entertaining and intriguing, and gave me insight into how truly new and innovative ideas emerge. When you walked into a lecture Mike gave, as with Forest Gump's box of chocolates, you never knew what you were gonna get, but you knew it would be good.

I also spent a lot of time in the UVic grad bar with Mike, both after class and work. These get-togethers had anywhere between three and nine people, but were always intimate. The back-and-forth of ideas was even more varied and playful than in the office meetings and courses, fueled in part by generous plates of nachos and jugs of Rickard's Red (which Mike insisted on paying for, saying it was his duty as a supervisor). Many of these ideas died (a much-loved proof of the collapse of the W -hierarchy to $W[2]$ lasted only 24 hours), but many also survived to appear later in print. Interspersed through it all was our realization that we were in the middle of something new and beautiful, and we wondered aloud (especially as the level of beer in the jugs lowered) when the rest of the world would see the parameterized light as we already had.

After all this time together, I got to know, appreciate, and occasionally puzzle over some of Mike's other interests outside of research. His passion for CS and Mathematics education was awe-inspiring. I spent many evenings helping out with this, both putting materials together beforehand (to this day I cannot look at rolls of hockey tape without remembering the hours we spent in the living room of Mike's house putting together neon-bright executable illustrations of sorting networks and graph problems on room-sized blue tarpaulins) and running the associated activities in school auditoriums in and around Victoria. I still have pictures of Mike encouraging children and their parents as they worked through these activities and discovered (without proof, but feeling their rightness) classical CS algorithms and complexity-theoretic distinctions. I heard of, but never experienced first-hand, his love of surfing, as he could never find a wet-suit big enough to fit me. This was perhaps fortunate. MikeH (who was wet-suitable) later told me about Mike's habit of, just as a wave you were trying to catch was getting interesting and hence potentially dangerous, starting distracting discussions on mathematical proofs.

If you hung around long enough, you got to glimpse Mike's loopier aspects. Sometimes they clung tenuously to the side of valid academia. I once narrowly talked Mike out of his brilliant idea of having me illustrate the finer points of parameterized analysis at an annual student-industry get-together in Vancouver by standing on a multicolored Rock of Complexity while wearing a clown suit. There were his surreal Passion Plays, written to bring home the beauty of various lesser-known branches of mathematics to the general public. Other times these aspects were part of his decidedly unconventional life. One day he brought in videotapes in which, over two sessions and about 7 1/2 hours, he told part of the story of how he volunteered for, went AWOL (several times) from, was imprisoned by, and was finally discharged (first dishonorably and then honorably) from the US Air Force during the Vietnam War. They were filmed by a cousin of his as working notes for a movie screenplay. They were amazing. Perhaps inevitably, they vanished from circulation after copies were given to several local schools. I wish I had kept one.

There were darker aspects as well. Mike can be both laid back and intense, personable and dispassionate. I think this is all part of what makes him an excellent and innovative mathematician. However, when unexpectedly combined, these aspects can be disconcerting. I remember a lunch-time meeting in which Mike evaluated an outline of one of my thesis chapters. He became more and uncomfortable trying to be nice about it until I gave him permission to stop being diplomatic, at which point he sighed, relaxed, and happily tore what I had written to shreds. There was a picture of Mike at that time in front of the CS General Office at UVic in which he looked directly at the camera with his usual smile and half his face was in perfect shadow. I felt then (and still feel) that there is truth in that picture.

Ultimately, though, it was good being around him. I was deeply impressed by Mike's generosity with ideas and his willingness to share authorship. As his students, we were always given the opportunity to become part of whatever papers Mike was working on. Perhaps even better was his not requiring that he be author on what we ourselves produced unless he contributed — if what we wrote got accepted, he would gladly pay to send us to meetings with single-author papers. I did not realize until years later just how special and unusual that was, and it is these things, among others above, that I carry forward.

Eventually, it came to an end. By the time my thesis was submitted, Mike was traveling a lot, on the verge of leaving Victoria for good, and I had taken a postdoc in Ontario. With the additional complication of an external examiner from South Africa, it was hard to arrange a defense date; at one point, we joked that it could only be held in a to-be-specified airport boarding lounge. However, it all came together in April 1999, 6 1/2 years after I first talked to Mike.

I'm faculty now, and it is the job of my dreams. I teach and have my own students, and enjoy both very much. Courtesy of my being one of Mike's early

graduate students, I've had the privilege of attending several of the parameterized complexity workshops at Dagstuhl. I see Mike at these workshops and he is as amazing and full of neat ideas and energetic as ever.

Looking back, Mike is the most fascinating person I have ever met and one of the greatest influences on my academic life. Being around him changed and gave form to my research, and his theories underlie much of my own work and intellectual outlook. Almost all of my research collaborators are people I have met either through Mike or by association with parameterized complexity. In my dealings with graduate students, I aim for his generosity and openness. In my teaching, to the best I can, I try to be passionate and convey to students the excitement in every subject that Mike does whenever he talks.

All told, pretty good results for one e-mail.

Happy birthday, Mike, and thank you. Please keep on thinking and doing beautiful things.

The Impact of Parameterized Complexity to Interdisciplinary Problem Solving

Ulrike Stege*

Department of Computer Science
University of Victoria
stege@cs.uvic.ca

Abstract. We discuss interdisciplinary parameterized complexity research in biology and cognitive science.

1 Introduction

Thinking back to my time as a PhD student in the Bioinformatics group at ETH Zürich, my first true encounter with Parameterized Complexity was when my colleague Chantal Korostensky and I followed an invitation from Mike Fellows to visit his research group in Victoria, British Columbia. We were both working on problems that were computationally hard—Chantal investigated methods to compute multiple sequence alignments [62] and I studied evolutionary tree reconciliation problems—and were fascinated hearing Mike’s novel ideas of how to deal with NP-hard problems without necessarily sacrificing optimality.

In the fall of 1997, Mike explained to us his vision to view NP-hard computational problems in a more refined way and challenged us to study these parameterized versions in a new framework, called *parameterized complexity* [27]. Using graph theoretic examples such as VERTEX COVER, INDEPENDENT SET and DOMINATING SET he illustrated that parameterization can provide a better understanding of why a problem, which is characterized as intractable in the first place, may not be truly intractable, and what aspect of the problem may or may not contribute to its intractability. All these foundational illustrations can be found in the famous book by Downey and Fellows, which was at the time almost completed [18].

When considering complexity in the classical sense, the time complexity of an algorithm is measured in the input size n of the problem input. That is, if a problem is identified to be NP-hard, there is likely no polynomial time algorithm. In other words, there is no algorithm with a running time $O(n^c)$ where c is a constant.

In contrast, in the parameterized world, the problem input is considered in terms of the input size n and a parameter k . If there exists an algorithm solving the problem with a running time of $O(f(k)n^c)$, then this parameterization of the problem is *fixed-parameter tractable* or a member of the parameterized

* Research supported by an NSERC Discovery Grant.

complexity class *FPT* [18]. $f(k)$ can be a super-polynomial time function; in this case, the “intractability” of the problem is “trapped,” that is, it depends on parameter k only and not on input size n . If $f(k)$ can be kept small for practical applications such that $f(k)$ is not “too big,” then the algorithm should behave just like a polynomial-time algorithm!

I was intrigued by this radical idea. The realization that the (in)tractability of hundreds of computational problems could be (re)considered, and that many of these problems might have efficient parameterized algorithms and therefore might not be that intractable—or not that impractical—after all, opened up a world of possibilities for me, and gave me ideas for plenty of PhD topics. Particularly, the attraction to consider parameterizations from the application perspective of the problem gave me new hope to escape heuristic approaches in interdisciplinary areas, such as bioinformatics.

This led me directly to the for me most convincing argument why we should continue to pursue parameterized complexity: namely because of the great potential of parameterized algorithms to be used to solve applied problems, by users from many different fields, including researchers in sciences and social sciences, as well as developers in industry.

My first aha-experience back then was that the NP-hardness property of a computational problem is by no means evidence for solid intractability but rather an encouragement to refine the problem statement using thoughtful parameterizations and to design parameterized algorithms. This was especially fascinating as it appeared to be counter to how I was taught and as I understood computational intractability at that time. Bodlaender *et al.* articulated this argument well in their 1995 article in *Computer Applications in the Biosciences* [5]. They argue that parameterized complexity (and not NP-completeness) is the appropriate tool for studying intractability. They further point out that parameterized intractability results can provide insights in possibly restricted versions of the problem as intractability results suggest (with respect to practical results) useful constraints.

Researchers who know Mike can easily guess that the first parameterized algorithm he showed to me to prove that a parameterized NP-complete problem can be in *FPT* was the bounded-search tree algorithm for VERTEX COVER with its promising practical running time of $O(2^k n)$ [21].¹ An other powerful method from the *FPT*-toolkit to apply to VERTEX COVER is efficient preprocessing, such as *kernelization* (cf. [9] for the first polynomial-time kernelization algorithm of VERTEX COVER). Kernelization preprocesses the input with guarantees, namely it reduces a problem instance to an equivalent problem instance whose size is bounded by some function of the parameter, or it solves the parameterized problem: In the case of VERTEX COVER parameterized by the size of the vertex cover to be determined, kernelization determines too large instances as no-instances [18]. For other parameterized problems, such as MAX LEAF SPANNING TREE parameterized by the number of leaves, kernelization reports large enough instances as yes-instances [28]. Especially desirable is polynomial-time

¹ Mehlhorn already described the basic idea of this elegant algorithm in 1984 [51].

kernelization. One of the most convincing arguments for the use of parameterized complexity in practice should be that a parameterized problem is in FPT if and only if it is kernelizable [19].

In general, preprocessing is a useful first step for solving any computational problem when dealing with large inputs or intractability properties. The beauty about VERTEX COVER is not just its presentability for all kinds of parameterized algorithmic techniques (e.g., [43,54,13]), and its impressive sequence of parameterized algorithms [21,9,58,18,17,2,19,55,70,69,12,13], but also its applicability—it is a powerful model for, for example, conflict graph resolution as it can be applied in post processing of gaps multiple sequence alignments [69] or in cleaning data of a character matrix when building phylogenetic trees (also called COMPATIBILITY [16]).

During my PhD studies, Mike provided me with many opportunities to meet and have discussions with biologists (e.g., Joe Felsenstein, University of Washington in Seattle, USA; my now colleague and collaborator Chris Upton, University of Victoria in British Columbia, Canada; Jack Heinemann, University of Canterbury in Christchurch, New Zealand, with whom we had most fascinating conversations about horizontal gene transfer) to explore how the message of parameterized complexity could be applied in sciences, and what collaboration with scientists could look like. He also introduced me to Todd Wareham who was finishing his PhD studies with Mike, and my future graduate student Iris van Rooij who introduced me to another favourite application area of mine, viz. Cognitive Psychology.² I fondly remember these memorable meetings and experiences.

The biggest challenge in these meetings was the research communication between the parties from the different areas: At the time, I became aware of the fact that academic groups develop their own particular science language, that is, for example, the “biology language” or the “computer science language,”³ and that an English sentence is often interpreted very differently by different scientific communities. For this reason alone, to succeed⁴ in interdisciplinary research, some of the necessary properties an interdisciplinary academic must possess are patience, the ability to question the (maybe existing) common grounds over and over again, and, therefore, one needs plenty of time to do well.

2 Parameterized Complexity and Interdisciplinary Research

A key during the problem solving process—when the goal is to design an algorithm or to provide even an implementation for an applied research problem—is

² Iris who had just finished her M.A. in Psychology was visiting Mike in Victoria during one of my later visits.

³ Some sciences have many “dialects” as my collaboration with people from networks and software engineering has demonstrated to me.

⁴ If you are bilingual, that is educated rigorously in both areas, then you can be a translator.

the ability to come up with a precise abstract problem description. Sometimes the algorithm designer may expect a formulation as a computational problem delivered by the user. However, formulating the problem can be challenging for the user. The algorithm designer must truly understand the problem at hand when making it abstract. Here, the translation between the different science languages is probably the most challenging part.

Even after a problem description is agreed upon, many times the first draft is just a stepping stone. To confirm that the right computational problem description has been found, often algorithmic results from real data inputs solving the problem are needed. If unsuccessful, the problem description must be revised. This process is part of the typical problem solving design cycle (such as the classic hypothetico-deductive method [83,84] used by scientists).

The task is not finished with the successful implementation of an algorithm. Practitioners typically rely on the integration of the implementation in a user-friendly and well-documented software package. Further, we want users to know what the advantages are of using optimal (or *exact*) algorithms compared to heuristics: for example, exact algorithms allow—in contrast to heuristics—an accurate evaluation of a computational problem when considered as a model of an applied research question. Therefore, our practical parameterized algorithms should be readily available for the user: we require user-friendly packages that combine parameterized and heuristic approaches and allow the user to opt for the exact approach whenever its running time is feasible. Efficient preprocessing in the form of data reduction should be applied as much as possible, even in cases where exact computation is impractical. Today, in many practical applications (such as sequence alignment problems), heuristic approaches are commonly applied to the underlying hard problems. One of the mission statements of applied parameterized complexity should be to offer additional exact approaches wherever possible as an alternative to heuristic approaches.

That parameterized algorithms can be practical is (still) best illustrated using the VERTEX COVER problem: VERTEX COVER instances containing a vertex cover of size up to at least $k = 200$ are considered practical. Best practices for implementations and use in bioinformatics for parameterized algorithms are the VERTEX COVER ones implemented in Langston's and Dehne's groups [42,44,43,15,11].

While the FPT-toolkit is great for solving computational problems algorithmically, parameterized complexity may also inform science when studying computational models in the search for models of *cognitive capacities* and cognitive processes [76,75]. *The tractability design cycle* is suggested as method to support this process.

We now turn our attention to the art of parameterization. For a given NP-hard computational problem, what parameterizations should be studied? While some parameter choices may seem natural, other choices appear less obvious. Of course, often the application can shed light on natural parameterizations that are not obvious otherwise. When, for example, parameterizing VERTEX COVER by the size of the cover to be determined (aka k -VERTEX COVER, the

natural parameterization of VERTEX COVER), the problem is in FPT, while when parameterizing by the number of vertices not to be included in the cover—aka k -INDEPENDENT SET, a parameterization of VERTEX COVER that is *dual* to k -Vertex Cover—the problem turns out to be complete for the class $W[1]$ [18]. If we choose as the parameter a less obvious (but also dual) variant, namely the number of edges covered minus the number of vertices in the cover (aka p -PROFIT COVER [71]), then the problem is again in FPT.

In terms of the practicality of FPT algorithms, parameters should be small to achieve acceptable running times [54]: that is, to solve the problem’s optimization version, the optimum value for the parameter, when solving the problem’s optimization version, should be small.⁵ Do the same limitations on the choice of parameters hold for all purposes of parameterized complexity? First, finding tractable parameterized algorithms for different parameterizations of a problem may improve practical running times [71]. Second, the application can shed light on the practicality of a parameter: If VERTEX COVER is used to model a conflict graph to clean a data set (c.f. [69]), then we expect k , the number of data points (vertices) to be removed from the data set (set of vertices of the graph), to be small as otherwise the data set can be considered as too noisy to be kept. We may want to use PROFIT COVER to verify that a data set is profitable (enough). Further, a complete parameterized analysis can be useful when for example evaluating computational models as models for cognitive theories [81,76]. When studying the cognitive limitation of humans in human problem solving, an exhaustive study of the complexity of possible parameterizations may prove informative. In particular, even tractable parameterized results that are considered trivial, such as parameterizations above or below the optimum solution [47], may be of interest when the goal is to evaluate possible cognitive theories as efficiently computable functions.

Niedermeier discusses the issue of how to parameterize in his book *Invitations to Fixed-Parameter Algorithms* from the perspective of an algorithm designer [54], and later distinguishes different ways of identifying parameters, namely solution quality driven ones (such as the size of a solution), and structural ones (parameterizing by distance from triviality, based on data analysis, by deconstructing hardness proofs, by dimensions,⁶ and averaging out) [53]. Parameterizations that depend on the real data sets may be most effective when developing a data-driven algorithmic implementation for the user. However, this way of parameterizing involves strong interdisciplinary collaboration: in [22], Fellows *et al.* say

“Identifying parameters relevant to real-world datasets is something of an art [53] and essential to the useful deployment of the multivariate outlook on NP-hard problems. In some sense, the search for relevant parameters brings this part of theoretical computer science to the fields of Heuristics and Algorithms Engineering and Artificial Intelligence.”

⁵ This discussion is closely related to the, in the early years discussed, *klam value* [18].

⁶ e.g., dimensions of input objects.

Niedermeier observes that the identification of parameterizations based on data analysis is still underdeveloped [53]. This might be one of the strongest expectations that is expected from interdisciplinary parameterized complexity research. A rigorous understanding of the data sets will yield an improved data-driven algorithm design-process by combining theoretical approaches with data facts.

3 Parameterized Complexity and Its Contributions to Computational Biology

Computational biology has received considerable attention from parameterized complexity researchers who have attacked a number of computational problems from the area. Early work considered the PERFECT PHYLOGENY problem [6], DNA Physical Mapping [26], and sequence or alignment problems [5]; variants of maximum agreement subtree problems were studied in [23,24]. Tree reconciliation problems for gene and species trees are studied in [25,68,69,35,3]. Over the years, many problems in computational biology have been considered in the parameterized framework. There is a number of PhD theses with significant contributions in this area, including [34,8,20,69,37,7,31,67,33,64], with Hallett's thesis being the first [34]. For a survey summarizing results on parameterized algorithms in phylogenetics we refer to the article by Gramm *et al.* [32]. Typical problem parameters in these problems include the number of input objects to be considered, the number of evolutionary events (following the parsimony assumption this number should be small), and combinations thereof.

Most parameterized complexity work in bioinformatics or computational biology is of theoretical nature. However, in addition to the practical work by the groups of Langston and Dehne mentioned above, Hüffner and his collaborators have done recent work in algorithmic engineering with focus on the bioinformatics area [38,36].

As discussed in Section 2, deep knowledge of the problem structure, the application domain and the properties of the real data sets are crucial for the design of algorithms and implementations driven by specific applications. Interdisciplinary collaboration and knowledge of both domains for all parties are useful if not necessary for success. It appears that most results in the intersection of parameterized complexity and computational biology are published in computer science conferences and journals, with some exceptions: examples are the work by Hallett *et al.* with their publication at the computational biology conference Recomb on identifying duplications and lateral gene transfers, van Brevern *et al.* with the recent article on motif search in *Transactions on Computational Biology and Bioinformatics* [4], and Hüffner *et al.* on clustering in *Biological Networks* [40]. The article *Developing Fixed-Parameter Algorithms to Solve Combinatorially Explosive Biological Problems* by Hüffner *et al.* published in *Bioinformatics* [39] is probably the one that best introduces the techniques of parameterized complexity to the biology community.

4 The Role of Parameterized Complexity in Cognitive Science

Human problem solving is a subarea of cognitive science that studies *human* problem solving in terms of problem solving strategies and performance, and looks for models of *cognitive capacities* (also denoted *computational-level theories* [48,76,75]) as well as cognitive processes. In the past, results from computational complexity have influenced this research,⁷ and were used in particular to justify the rejection of computational problems as potential computational-level theories according to what van Rooij denoted the *P-cognition thesis* [75]: only computational problems that can be solved in polynomial time can serve as computational-level theories. That is, computational problems that are shown to be NP-hard were either rejected or inexact solution strategies were suggested as explanations for people dealing with the intractability of these problems (e.g., approximation algorithms or heuristic). Wareham was the first arguing in his 1996 paper that parameterized complexity is the better tool than classical computational complexity when measuring the complexity of computational models for cognitive systems or identifying the sources that isolate the model power [80]. In recent years, the consideration of parameterized complexity has led to a relaxation of the P-cognition thesis, resulting in the *FPT-cognition thesis*: NP-hard but fixed-parameter tractable parameterized computational decision problems can also be considered candidates for computational-level theories [76,75].

Modelling Cognitive Capacities. A common assumption amongst a group of cognitive scientists is that cognitive capacities consist of input/output functions that are “efficiently computable” (cf. the works by van Rooi [76,75] as well as articles by Cherniak [14], Frixione [29], Levesque [45] and Tsotsos [73]). Further, it is widely assumed that computational complexity can aid cognitive science research [1,49,50,52,56,57,61,65,66,73]. As Tsotsos noted in his author’s response to the commentaries of his article from 1990, complexity analysis is an important dimension of study when modelling vision [73].

Typical questions when investigating cognitive capacities and studying human problem solving include: What computational problems are good candidates for models of cognitive capacities? How do people solve these computational problems? What are people’s limitations w.r.t. instance sizes and properties? What are plausible models for the human solution process?

Many computational problems that are discussed in the literature as candidates for models of cognitive capacities are characterized as NP-hard [66,73,59,46,80,50,81,30,72,76,79,78,77,82]. As a consequence, to serve as models for cognitive capacities, these problems are either disregarded in their general form, or they decoy the researcher to assume that people use approximation algorithms or heuristics to solve the tasks. Complexity analysis as such has led to criticism of its relevance for this purpose (cf. commentaries to [73]). The use of parameterized complexity in this area has shed a different light on problems that are NP-hard but have parameterizations that are members of the

⁷ The dissertations by Wareham and van Rooij are examples surveying this literature.

class FPT and thus put the plausibility of computational problems as cognitive models into a different perspective: since there exist exact algorithms that are tractable for some NP-hard problems, heuristics and approximation algorithms may not be the only possible explanation for approaches that humans choose when tackling certain problems.

Parameterized complexity was first suggested by Wareham as a tool to evaluate models of cognitive systems [80]. He argues that a refined analysis using parameterized complexity proves more useful than classical complexity analysis alone. To evaluate an NP-hard problem systematically as a model of a cognitive capacity—that is, to fully characterize a computational problem by showing what and what does not make it tractable—Wareham suggest to use what he calls *systematic parameterized complexity analysis* [81].

Some of the first discussed computational problems as models of cognitive capacities using parameterized complexity were phonological models in linguistics [80,81]—namely *Declarative Phonology* problems [63] and *Optimality Theory* [60]. Both are constraint-based theories: Optimality Theory has a priority order for constraint satisfaction while in Computational Declarative Phonology problems all constraints must be satisfied; described are rule based phonological mechanisms that manipulate the (mental or spoken) phonological representation. Wareham discusses two distinct processes for a *declarative phonology system* $S = \langle P, D \rangle$, that is a pair of constraint sets where P encodes the phonological mechanisms of a language and D encodes the lexicon of the language: *encoding* with its corresponding computational problem DP-ENCODE and *decoding* with DP-DECODE [81]. A systematic parameterized complexity analysis over a spectrum of input and solution driven parameterizations for both theories reveals several tractable parameterizations that may be investigated further as plausible models in linguistics [81].

Van Rooij *et al.* studied Subset Choice problems that model decision making tasks [76,78]. Subset choice problems can arise in a variety of settings: choice situations in everyday settings (e.g., when selecting toppings for a pizza) as well as highly specialized ones (e.g., prescribing medication). Different models of the task of choosing a subset of items from among a set of available items are investigated [76,78].

Most recent work includes the discussion of the complexity of self-organization of cognitive behaviour using CONSTRAINT SATISFACTION by van Rooij [74], on Bayesian Intractability by Kwisthout *et al.* [41] and the human performance of *Vertex Cover* by Carruthers *et al.* [10].

5 Conclusions

Many hard problems in practical applications exhibit a rich structure that allows a number of realistic parameterizations that permit the design of parameterized algorithms. In bioinformatics, there is a tremendous need for developing exact algorithms for problems with very large data inputs. To move even more of the highly evolved algorithmic results from parameterized complexity in bioinformatics towards sophisticated practical implementations, the area will profit

from: continued interdisciplinary collaboration, a push in parameterized algorithmic engineering, and a series of publications in the biology community to publicize the methods and enlighten an even larger number of researchers about the advantages of optimal algorithms compared to heuristics. To succeed, from the applied user's standpoint, more practical parameterized algorithms should be readily available.

The research findings in cognitive modeling have the potential to significantly impact the ability of cognitive psychologists to identify new theories that model cognitive capacities and problem solving processes. In contrast to the computational biology field, the majority of parameterized complexity research in cognitive science is published in cognitive science and psychology journals. In particular van Rooij has begun to publicize findings in the cognitive science community.⁸

While parameterized complexity results in computational biology mainly focus on the development of fast algorithms, in cognitive psychology the main focus is on the modelling of cognitive capacities. Computational biology can learn from the the modeling research done in cognitive science to improve the process of formalizing of computational problems.

Acknowledgements. I want to thank Hausi Müller and my research group at UVic, in particular Sarah Carruthers, for lots of interesting discussions on the subject.

References

1. Anderson, J.R.: The Adaptive Character of Thought. Lawrence Erlbaum Associates (1990)
2. Balasubramanian, R., Fellows, M.R., Raman, V.: An Improved Fixed-Parameter Algorithm for Vertex Cover. *Information Processing Letters* 65, 163–168 (1998)
3. Bansal, M.S., Shamir, R.: A Note on the Fixed Parameter Tractability of the Gene-Duplication Problem. *IEEE/ACM Trans. Comput. Biol. Bioinformatics* 8(3), 848–850 (2011)
4. Betzler, N., van Bevern, R., Komusiewicz, C., Fellows, M.R., Niedermeier, R.: Parameterized Algorithmics for Finding Connected Motifs in Biological Networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 8(5), 1296–1308 (2011)
5. Bodlaender, H.L., Downey, R.G., Fellows, M.R., Hallett, M.T., Wareham, H.T.: Parameterized Complexity Analysis in Computational Biology. *Computer Applications in the Biosciences* 11(2), 49–57 (1995)
6. Bodlaender, H.L., Fellows, M.R., Warnow, T.J.: Two Strikes Against Perfect Phylogeny. In: Kuich, W. (ed.) *ICALP 1992*. LNCS, vol. 623, pp. 273–283. Springer, Heidelberg (1992)
7. Boucher, C.A.: *Combinatorial and Probabilistic Approaches to Motif Recognition*. PhD thesis, University of Waterloo (2010)

⁸ Parameterized complexity was used by several participants in their presentations at the recent interdisciplinary Seminar *Computer Science & Problem Solving: New Foundations* that took place Dagstuhl in 2011.

8. Bryant, D.: Building Trees, Hunting for Trees, and Comparing Trees—Theory and Methods in Phylogenetic Analysis. PhD thesis, University of Canterbury (1997)
9. Buss, J.F., Goldsmith, J.: Nondeterminism within P. *SIAM J. Comput.* 22(3), 560–572 (1993)
10. Carruthers, S., Masson, M., Stege, U.: Human Performance on Hard Non-Euclidean Graph Problems: Vertex Cover. Accepted to the *Journal of Problem Solving* (2012)
11. Cheetham, J., Dehne, F., Rau-Chaplin, A., Stege, U., Taillon, P.J.: Solving Large FPT Problems on Coarse Grained Parallel Machines. *Journal of Computer and System Sciences* 67(4), 691–706 (2003)
12. Chen, J., Kanj, I.A., Jia, W.: Vertex Cover: Further Observations and Further Improvements. In: Widmayer, P., Neyer, G., Eidenbenz, S. (eds.) *WG 1999*. LNCS, vol. 1665, pp. 313–324. Springer, Heidelberg (1999)
13. Chen, J., Kanj, I.A., Xia, G.: Improved Parameterized Upper Bounds for Vertex Cover. In: Kráľovič, R., Urzyczyn, P. (eds.) *MFCS 2006*. LNCS, vol. 4162, pp. 238–249. Springer, Heidelberg (2006)
14. Cheriak, C.: *Minimal Rationality*. MIT Press (1986)
15. ClustalXP, <http://clustalxp.cgmlab.org/> (retrieved 2012)
16. Day, W.H.E., Sankoff, D.: Computational Complexity of Inferring Phylogenies by Compatibility. *Syst. Zool.* 35(2), 224–229 (1986)
17. Downey, R.G., Fellows, M.R.: Fixed parameter tractability and completeness II: Completeness for $W[1]$. *Theoretical Computer Science A* 141, 109–131 (1995)
18. Downey, R.G., Fellows, M.R.: *Parameterized Complexity*. Springer (1999)
19. Downey, R.G., Fellows, M.R., Stege, U.: Parameterized Complexity: a Framework for Systematically Confronting Computational Intractability. In: Graham, R., Kratochvíl, J., Nešetřil, J., Roberts, F. (eds.) *Proc. DIMACS-DIMATIA Workshop, Prague*. *Contemporary Trends in Discrete Mathematics* (1997), *AMS-DIMACS Series in Discrete Mathematics and Theoretical Computer Science* 49, 49–99 (1999)
20. Evans, P.: *Algorithms and Complexity for Annotated Sequence Analysis*. PhD thesis, University of Victoria (1999)
21. Fellows, M.R.: On the Complexity of Vertex Set Problems. Technical Report, Computer Science Department, University of New Mexico (1988)
22. Fellows, M.R., Gaspers, S., Rosamond, F.: Multivariate Complexity Theory. In: Blum, E.K., Aho, A.V. (eds.) *Computer Science*, pp. 269–293. Springer, New York (2011)
23. Fellows, M.R., Hallett, M.T., Korostensky, C., Stege, U.: Analogs and Duals of the MAST Problem for Sequences and Trees. In: Bilardi, G., Pietracaprina, A., Italiano, G.F., Pucci, G. (eds.) *ESA 1998*. LNCS, vol. 1461, pp. 103–114. Springer, Heidelberg (1998)
24. Fellows, M.R., Hallett, M.T., Stege, U.: Analogs and Duals of the MAST Problem for Sequences and Trees. *Journal of Algorithms* 49(1), 192–216 (2003)
25. Fellows, M.R., Hallett, M.T., Stege, U.: On the Multiple Gene Duplication Problem. In: Chwa, K.-Y., Ibarra, O.H. (eds.) *ISAAC 1998*. LNCS, vol. 1533, pp. 347–357. Springer, Heidelberg (1998)
26. Fellows, M.R., Hallett, M.T., Wareham, H.T.: DNA Physical Mapping: Three Ways Difficult. In: Lengauer, T. (ed.) *ESA 1993*. LNCS, vol. 726, pp. 157–168. Springer, Heidelberg (1993)
27. Fellows, M.R., Langston, M.: Nonconstructive Advances in Polynomial Time Complexity. *Information Processing Letters* 26, 157–162 (1987)

28. Fellows, M.R., McCartin, C., Rosamond, F.A., Stege, U.: Coordinatized kernels and Coordinatized Kernels and Catalytic Reductions: An Improved FPT Algorithm for Max Leaf Spanning Tree and Other Problems. In: Kapoor, S., Prasad, S. (eds.) FST TCS 2000. LNCS, vol. 1974, pp. 240–251. Springer, Heidelberg (2000)
29. Frixione, M.: Tractable competence. *Minds and Machines* 11, 379–397 (2001)
30. Graham, S.M., Joshi, A., Pizlo, Z.: The Traveling Salesman Problem: A Hierarchical Model. *Memory & Cognition* 28(7), 1191–1204 (2000)
31. Gramm, J.: Fixed-Parameter Algorithms for the Consensus Analysis of Genomic Data. PhD thesis, Universität Tübingen (2003)
32. Gramm, J., Nickelsen, A., Tantau, T.: Fixed-Parameter Algorithms in Phylogenetics. *The Computer Journal* 51(1), 79–101 (2008)
33. Guo, J.: Algorithm Design Techniques for Parameterized Graph Modification Problems. PhD thesis, Friedrich-Schiller-Universität Jena (2006)
34. Hallett, M.T.: An Integrated Complexity Analysis of Problems from Computational Biology. PhD thesis, University of Victoria (1996)
35. Hallett, M., Lagergren, J., Tofight, A.: Simultaneous Identification of Duplications and Lateral Transfers. In: RECOMB 2004, pp. 47–356 (2004)
36. Helwig, S., Hüffner, F., Rössling, I., Weinard, M.: Chapter 3. Selected Design Issues. In: Müller-Hannemann, M., Schirra, S. (eds.) *Algorithm Engineering*. LNCS, vol. 5971, pp. 58–126. Springer, Heidelberg (2010)
37. Hermelin, D.: New Results in Parameterized Complexity. PhD thesis, University of Haifa (2009)
38. Hüffner, F.: Algorithm Engineering for Optimal Graph Bipartization. *Journal of Graph Algorithms and Applications* 13(2), 77–98 (2009)
39. Hüffner, F., Niedermeier, R., Wernicke, S.: Developing Fixed-Parameter Algorithms to Solve Combinatorially Explosive Biological Problems. *Bioinformatics*, 395–421 (2007)
40. Hüffner, F., Niedermeier, R., Wernicke, S.: Fixed-parameter algorithms for graph-modeled data clustering. In: *Clustering Challenges in Biological Networks*, pp. 3–28. World Scientific (2009)
41. Kwisthout, J., Wareham, T., van Rooij, I.: Bayesian Intractability is not an Ailment that Approximation Can Cure. *Cognitive Science* 35(5), 779–784 (2011)
42. Langston, M.A.: Homepage, <http://web.eecs.utk.edu/~langston/> (retrieved 2012)
43. Langston, M.A., Abu-Khzam, F.N., Collins, R.L., Fellows, M.R., Suters, W.H., Symons, C.T.: Kernelization Algorithms for the Vertex Cover Problem: Theory and Experiments. In: *ALLENEX 2004*, pp. 62–69 (2004)
44. Langston, M.A., Abu-Khzam, F.N., Shanbhag, P.: Scalable Parallel Algorithms for Difficult Combinatorial Problems: a Case Study in Optimization. In: *PDCS 2003*, pp. 649–654 (2003)
45. Levesque, H.J.: Logic and the complexity of reasoning. *Journal of Philosophical Logic* 17, 355–389 (1988)
46. MacGregor, J.N., Ormerod, T.C., Chronicle, E.P.: A Model of Human Performance on the Traveling Salesperson Problem. *Memory & Cognition* 28(7), 1183–1190 (2000)
47. Mahajan, M., Raman, V., Sikdar, S.: Parameterizing Above or Below Guaranteed Values. *J. Comput. Syst. Sci.* 75(2), 137–153 (2009)
48. Marr, D.: *Vision: A Computational Investigation into the Human Representation and Processing Visual Information*. W.H. Freeman (1982)
49. Martignon, L., Hoffrage, U.: Fast, Frugal, and Fit: Simple Heuristics for Paired Comparison. *Theory and Decision* 52, 29–71 (2002)

50. Martignon, L., Schmitt, M.: Simplicity and Robustness of Fast and Frugal Heuristics. *Minds and Machines* 9, 565–593 (1999)
51. Mehlhorn, K.: Data Structures and Efficient Algorithms. *Graph Algorithms and NP-Completeness*, vol. 2. Springer (1984)
52. Millgram, E.: Coherence: The Price of the Ticket. *Journal of Philosophy* 97, 82–93 (2000)
53. Niedermeier, R.: Reflections on Multivariate Algorithmics and Problem Parameterization. In: *STACS 2010*, pp. 17–32 (2010)
54. Niedermeier, R.: *Invitation to Fixed-Parameter Algorithms*. Oxford University Press (2006)
55. Niedermeier, R., Rossmanith, P.: Upper Bounds for Vertex Cover Further Improved. In: Meinel, C., Tison, S. (eds.) *STACS 1999*. LNCS, vol. 1563, pp. 561–570. Springer, Heidelberg (1999)
56. Oaksford, M., Chater, N.: Reasoning Theories and Bounded Rationality. In: Manktelow, K.I., Over, D.E. (eds.) *Rationality: Psychological and Philosophical Perspectives*, pp. 31–60 (1998)
57. Oaksford, M., Chater, N.: *Rationality in an uncertain world: Essays on the cognitive science of human reasoning*. Psychology Press, Hove (1993)
58. Papadimitriou, C.H., Yannakakis, M.: On limited nondeterminism and the complexity of the V-C dimension. *Journal of Computer and System Sciences* 53(2), 161–170 (1996)
59. Parberry, I.: Knowledge, Understanding, and Computational Complexity. In: Levine, D.S., Elsberry, W.R. (eds.) *Optimality in Biological and Artificial Networks?*, pp. 125–144. Lawrence Erlbaum Publishers, Hillsdale (1997)
60. Prince, A., Smolensky, P.: *Optimality Theory Constraint Interaction in Generative Grammar*. Tech. Rep. RuCCS TR-2, Center for Cognitive Science, Rutgers University (1993)
61. Rensink, R.A., Provan, G.: The Analysis of Resource-Limited Vision Systems. In: *Proceedings of the 13th Annual Conference of the Cognitive Science Society*, pp. 311–316 (1991)
62. Roth-Korostensky, C.: *Algorithms for Building Multiple Sequence Alignments and Evolutionary Trees*. PhD thesis, ETH Zürich (2000)
63. Scobbie, J.M.: *Towards Declarative Phonology*. In: Bird, S. (ed.) *Declarative Perspectives in Phonology*. Edinburgh Working Papers in Cognitive Science, vol. 7, pp. 1–27. University of Edinburgh (1992)
64. Shaw, P.: *Advances in Cluster Editing: Linear FPT Kernels and Comparative Implementations*. PhD thesis, The University of Newcastle (2010)
65. Simon, H.A.: Rationality as Process and as Product of Thought. In: Bell, D.E., Raiffa, H., Tversky, A. (eds.) *Decision Making: Descriptive, Normative, and Prescriptive Interactions*, pp. 58–77. Cambridge University Press, Cambridge (1988)
66. Simon, H.A.: Invariants of human behavior. *Annual Review of Psychology* 41(1), 1–19 (1990)
67. Snir, S.: *Computational Issues in Phylogenetic Reconstruction: Analytic Maximum Likelihood Solutions, and Convex Recoloring*. PhD thesis, Technion (2004)
68. Stege, U.: *Gene Trees and Species Trees: The Gene-Duplication Problem is Fixed-Parameter Tractable*. In: Dehne, F., Gupta, A., Sack, J.-R., Tamassia, R. (eds.) *WADS 1999*. LNCS, vol. 1663, pp. 288–293. Springer, Heidelberg (1999)
69. Stege, U.: *Resolving Conflicts from Computational Biology*. PhD thesis, ETH Zürich (2000)

70. Stege, U., Fellows, M.R.: An Improved Fixed-Parameter Tractable Algorithm for Vertex Cover. Technical Report 318, Department of Computer Science, ETH Zürich (April 1999)
71. Stege, U., van Rooij, I., Hertel, A., Hertel, P.: An $O(pn + 1.151^p)$ -Algorithm for p -Profit Cover and Its Practical Implications for Vertex Cover. In: Bose, P., Morin, P. (eds.) ISAAC 2002. LNCS, vol. 2518, pp. 249–261. Springer, Heidelberg (2002)
72. Thagard, P.: Coherence in Thought and Action. MIT Press (2000)
73. Tsotsos, J.K.: Analyzing Vision at the Complexity Level. Behavioral and Brain Sciences 13(3), 423–469 (1990)
74. Rooij, I.: Self-Organization Takes Time too. Topics in Cognitive Science 4, 63–71 (2012)
75. van Rooij, I.: The tractable Cognition Thesis. Cognitive Science (2008)
76. van Rooij, I.: Tractable Cognition: Complexity Theory in Cognitive Psychology. Ph.D. Thesis, Department of Psychology, University of Victoria (2003)
77. van Rooij, I., Schactman, A., Kadlec, H., Stege, U.: Perceptual or Analytical Processing? Evidence from Children’s and Adult’s Performance on the Euclidean Traveling Salesman Problem. Journal of Problem Solving 1(1), 44–73 (2006)
78. van Rooij, I., Stege, U., Kadlec, H.: Sources of Complexity in Subset Choice. Journal of Mathematical Psychology 49(2), 160–187 (2005)
79. van Rooij, I., Stege, U., Schactman: Convex hull and Tour Crossings in the Euclidean Traveling Salesperson Problem: Implications for Human Performance Studies. Memory & Cognition 31(2), 215–220 (2003)
80. Wareham, H.T.: The role of Parameterized Computational Complexity Theory in Cognitive Modeling. In: AAAI 1996 Workshop Working Notes: Computational Cognitive Modeling: Source of the Power (1996)
81. Wareham, H.T.: Systematic Parameterized Complexity Analysis in Computational Phonology. Ph.D. Thesis, Department of Computer Science, University of Victoria (1999)
82. Wareham, H.T., Evans, P., van Rooij, I.: What does (and doesn’t) make analogical problem solving easy? Journal of Problem Solving 3(2), 30–71 (2011)
83. Whewell, W.: History of the Inductive Sciences, from the Earliest to the Present Times, London, vol. 3 (1837)
84. Whewell, W.: The Philosophy of the Inductive Sciences, founded upon their history, London, vol. 2 (1840)

Vertex Cover, Dominating Set and My Encounters with Parameterized Complexity and Mike Fellows

Venkatesh Raman

The Institute of Mathematical Sciences,
Chennai, India 600 113
vraman@imsc.res.in

Abstract. In this report, I start with a historic view of how, the two problems VERTEX COVER and DOMINATING SET that were influential to the birth of the area of parameterized complexity, also led me to this area and introduced me to Mike Fellows. I also discuss early research and meetings in Parameterized Complexity, Mike's influence in community building and some personal anecdotes with Mike. I conclude with some recent results on these two problems and also discuss open problems in the area.

1 Introduction (How It All Started for Me!)

Having done my PhD on Sorting algorithms, I took a natural liking to the family of directed graphs called tournaments, as a transitive tournament exactly models a totally ordered set. I started looking at papers [2] that showed connections between sorting algorithms and finding directed hamiltonian paths in tournaments. With some colleagues, I wrote a couple of papers describing efficient algorithms and lower bounds to find vertices with specific degrees in tournaments. Along the way, I stumbled into the paper [10] that discussed the complexity of dominating set in tournaments. From the paper, I learnt that every tournament on n vertices has a dominating set of size at most $\lceil \log n \rceil$, and I found it interesting that there are tournaments [6] on n vertices, constructed using simple number theoretic properties where the *minimum* dominating set size is $\Omega(\log n)$.

Using these properties (particularly the existence of tournaments with large dominating sets), I managed to give a reduction from dominating set in general directed graphs to dominating set in tournaments.. The reduction, pretty much, preserved the parameter (k went to $k+1$), but it used $O(2^k n)$ additional vertices and time. Essentially I had proved that the dominating set problem is $W[2]$ -hard in tournaments [14], though, at that time, I didn't know the terminology of W -hardness.

This led me to the paper by Papadimitriou and Yannakakis [13] where this problem was discussed in the context of needing limited non-determinism. There, I got the notions of parameterized complexity (in the last paragraph) and $O^*(4^k)$

algorithm for VERTEX COVER (Theorem 5 in that paper). I could quickly improve this to $O^*(2^k)$ through a different algorithm. With these two results ($W[2]$ -hardness result for dominating set in tournaments and $O^*(2^k)$ algorithm for Vertex Cover), I sent a mail to Mike Fellows explaining what I had proved.

The response I got was initially dampening, as Mike cited his paper [4] with Rod Downey, where both these results (Theorem 2.1 and Theorem 4.1) had already been proved. However, the response also opened me to the exciting world of parameterized complexity and got me in touch with Mike.

Later, Mike followed up by sending me a lot of his surveys, and some excerpts of the book he was writing with Rod Downey. He also invited me to visit him in Victoria if I was interested in working in the area. I jumped at the offer as I was scheduled to visit Waterloo, Canada that summer (of 1995) anyway.

In his response, Mike also asked whether 2^k for the $f(k)$ for VERTEX COVER can be improved further. I discussed this with my colleague R. Balasubramanian (who is the current director of our institute) and together we could improve the 2^k function to c^k for some constant $c < 2$. After my visit to Victoria, and discussions with Mike, we could exhibit a much improved c (close to 1.32) in c^k [1] which was the beginning of the long race of improvements for VERTEX COVER. Subsequently, I was quite happy to start the races for MAXSAT, and UNDIRECTED FEEDBACK VERTEX SET. Jianer Chen's group at Texas, and Rolf Niedermeier's group at Jena were the early participants in these races.

2 Early Meetings in Parameterized Complexity

I visited Mike again at Victoria two years later, and there had been big changes in his life since my previous visit (but one could hardly notice anything about them in discussions with him). I also visited him at New Zealand (on the way to COCOON at Australia to present our paper [8]) and there I got to meet Fran. Also during that visit¹ we worked on irredundant sets [5] and planar directed feedback vertex set (with not much progress on the later problem).

During those times, Mike was giving invited talks at various conferences and workshops and his infectious enthusiasm for giving talks caught my attention. I gave talks in this area at various places including the University of Waterloo, Canada, Max Planck Institute for Informatik, Saarbrücken, National Seminar [15] in India and IMSc. In fact our work on parameterizing above the guarantee [9] grew out of a question asked by someone during one of my talks. He felt cheated when I was showing that the standard parameterized question for MaxSAT was fixed-parameter tractable without giving a new algorithm.

While most of the parameterized complexity community maybe aware of the first meeting on parameterized complexity at Chennai in December 2000 (Mike mentions about this in the preface of the first IWPEC proceedings), not

¹ Interestingly, I landed up in a court with Mike and Fran in Wellington, as a witness to help them; the subtenant, to whom they had rented their apartment during their visit to Canada, refused to vacate and Mike and Fran had to settle that in the court; coincidentally the judge in the case was an Indian!

many would know about the ‘mini-symposium’ on parameterized complexity that Mike organized at the SIAM Discrete Mathematics conference at Toronto (www.siam.org/meetings/dm98/ms6.htm) in which some of us participated and gave talks. This is a flagship conference for Discrete Mathematics, and it is usually attended by big names in the field.

The first ‘workshop’ on Parameterized Complexity at IMSc Chennai in 2000 was organized at a short notice. This was organized largely by Mike and Fran during their first visit to India. Apart from a set of excellent talks (including by Jochen Alber, Marco Cesati and Liezhen Cai to name a few), the highlight of the meeting was a ‘problem solving’ workshop and an ‘auto-rickshaw’ trip to the Chennai beach, all organized by Mike and Fran.

Then again in 2002, we had a meeting on parameterized complexity as a pre-conference workshop to FSTTCS 2002 at IIT Kanpur. The big news at that time was the polynomial time algorithm for primality that came from IIT Kanpur. The first Dagstuhl workshop on parameterized complexity was eventful with the dissection of the new lower bound result by Cai and Juedes [3] that had appeared at that time. The series of IWPEC (which later became IPEC) workshops was born in that Dagstuhl meeting.

3 Conclusions

Parameterized complexity is a paradigm whose time has arrived. The large number of papers in almost all algorithms conference is a witness to this fact. While some of us may have been anguished by the lack of speed at which the area has penetrated among the theory community, even the current spread would not have been possible if not for Mike’s vigorous campaign, in all possible platforms, the practical uses of the paradigm, his efforts in community building at various places and his eagerness to encourage and work with anyone interested in the field.

Let me conclude by saying a few words about Mike and then with some recent developments on both the problems – VERTEX COVER and DOMINATING SET.

3.1 Mike Fellows

An article dedicated to Mike’s birthday can not do justice if it doesn’t mention Mike’s generosity. Mike’s generosity in readily sharing his ideas, perceptions, open problems, and his time and even money (even though he was living ‘on the edge’ most of the time) are well known. During my visits to Victoria and in some of his sessions in Chennai, I could see his deep passion for popularizing Mathematics. And I have seen first hand some of his Mathematics ‘fairs’ at Victoria. In Chennai, we could get to see the other side of Mike and Fran, as they were quite popular among the vendors, helpers, waiters and the auto rickshaw drivers, as their compassion to them knew no bounds. Mike’s breadth and passion for science in general was quite visible when he was rubbing shoulders with the physicists and mathematicians at our institute during his various visits. He was

quite an enthusiastic player in the team that did a review of our entire institute, a few years ago. Personally I have been inspired by many of his qualities, and let me use this occasion to say, ‘Thank you and Happy birthday Mike!’.

3.2 Recent Work on (above Matching Guarantee) Vertex Cover

A brief look at the table of FPT races [12] shows that VERTEX COVER continues to be one of the very few problems having an FPT running time $< 2^k$ (not including problems that have subexponential algorithms on special classes of graphs) which grew out of a question by Mike quite early on. For example, the running time of the closely related feedback vertex set problem in undirected graphs, is still above $O^*(3^k)$ and it required a new technique (iterated compression) to even reach this stage.

Recent attempts [11] to use linear programming to get improved parameterized algorithms for VERTEX COVER ABOVE THE MATCHING SIZE (AGVC) could pave for a new direction and race, to get improved parameterized algorithms for VERTEX COVER as well as for other problems. I am also glad that the ‘above guarantee parameterization’ is not just a natural paradigm (that took a life of its own), but AGVC in particular, has become a central parameterized problem to which several other natural problems could be reduced [11].

3.3 Recent Work on Dominating Set (in Graphs with Excluded Subgraphs)

The W -hardness proof of dominating set in tournaments continues to be one of the few parameterized reductions that is NOT a polynomial time reduction, but simple enough to be done in a first lecture on parameterized reductions.

On the algorithmic front on dominating sets, while a number of fixed parameter algorithms are known for planar, bounded genus and bounded treewidth graphs, I’d like to point to a (perhaps not so well known) result that the problem is fixed-parameter tractable in graphs having no ‘short’ cycles [16]. This has been generalized to graphs not having $K_{i,j}$ as a subgraph for fixed values of i and j , and this result eventually led to a polynomial kernel for bounded degenerate graphs [7]. These continue to be one of the few classes of graphs that are characterized by ‘forbidden subgraphs’ (as opposed to forbidden minors) for which fixed parameter algorithms are known for the dominating set problem.

3.4 Open Problems

I end with a couple of concrete problems that are still open from our first workshop on Parameterized complexity in Chennai. What is the parameterized complexity of the following problems (here k is the parameter)?

1. Given an undirected graph G and an integer k , does G have a complete bipartite graph as an induced subgraph with k vertices in each part?
2. Given an undirected planar graph G and an integer k , does G have an independent set on $n/4 + k$ vertices?

References

1. Balasubramanian, R., Fellows, M., Raman, V.: An improved fixed parameter algorithm for Vertex Cover. *Information Processing Letters* 65, 163–168 (1998)
2. Bar-Noy, A., Naor, J.: Sorting, minimal feedback sets, and Hamiltonian paths in Tournaments. *SIAM J. Discrete Mathematics* 3(1), 7–20 (1990)
3. Cai, L., Juedes, D.: On the existence of subexponential parameterized algorithms. *Journal of Computer and Systems Sciences* 67(4), 789–807 (2003); preliminary version in ICALP 2001
4. Downey, R., Fellows, M.: Parameterized Computational Feasibility. In: Clote, P., Remmel, J. (eds.) *Proceedings of the Second Cornell Workshop on Feasible Mathematics, Feasible Mathematics II*, pp. 219–244. Birkhauser, Boston (1995)
5. Downey, R., Fellows, M.R., Raman, V.: The complexity of irredundant sets parameterized by size. *Discrete Applied Mathematics* 100, 155–167 (2000)
6. Graham, R.L., Spencer, J.H.: A constructive solution to a tournament problem. *Canadian Mathematics Bulletin* 14, 45–48 (1971)
7. Philip, G., Raman, V., Sikdar, S.: A polynomial kernel for dominating set in $K_{i,j}$ -free and d -degenerate graphs. To appear in *ACM Transactions on Algorithms*; a preliminary version in *Proceedings of ESA 2009*
8. Khot, S., Raman, V.: The Parameterized complexity of finding subgraphs with hereditary properties. *Theoretical Computer Science* 289, 997–1008 (2002); a preliminary version appeared in *Proceedings of COCOON 2000*
9. Mahajan, M., Raman, V.: Parameterizing above the guarantee: maxsat and maxcut. *Journal of Algorithms* 31, 335–354 (1999)
10. Megiddo, N., Vishkin, U.: On finding a minimum dominating set in a tournament (Note). *Theoretical Computer Science* 61(2-3), 307–316 (1988)
11. Lokshtanov, D., Narayanaswamy, N.S., Raman, V., Ramanujan, M.S., Saurabh, S.: Faster Parameterized Algorithms using Linear Programming, <http://arXiv.org/abs/1203.0833>; preliminary version appeared as LP can be a cure for parameterized algorithms. In: *The Proceedings of STACS 2012*
12. Table of FPT races, <http://fpt.wikidot.com/fpt-races>
13. Papadimitriou, C.H., Yannakakis, M.: On Limited non-determinism and the complexity of the V-C Dimension. *Journal of Computer and Systems Sciences* 53(2), 161–170 (1996)
14. Raman, V.: Some hard problems in (weighted) tournaments. In: *Proceedings of the Fifth National Seminar on Theoretical Computer Science, Bombay*, pp. 115–122 (1995)
15. Raman, V.: Parameterized Complexity. In: *Proceedings of the Seventh National Seminar on Theoretical Computer Science, Chennai*, pp. I1–I18 (June 1997)
16. Raman, V., Saurabh, S.: Short cycles make W -hard problems hard; FPT algorithms for hard problems in graphs with no short cycles. *Algorithmica* 52(2), 203–225 (2008); preliminary version in the *Proceedings of SWAT 2006*

Mike Fellows: Weaving the Web of Mathematics and Adventure

Jan Arne Telle

Department of Informatics, University of Bergen, Norway

Abstract. This informal tribute in honor of Mike Fellows' 60th birthday is based on some personal recollections.

Mike Fellows is one of the founding fathers of parameterized complexity and among the few mathematicians who have really shaped theoretical computer science. This informal tribute in honor of his 60th birthday will mention a few of his lesser known deeds. It is based on personal recollections and will not even attempt to do justice to Mike's broad influence.

Michael Ralph Fellows was born near Los Angeles in California on 15th June, 1952. His family name derives from the Old Norse word *félagi*, mentioned on runic inscriptions in the meaning of comrade or weapon brother. As a young man he quite literally lived up to this nominative feature when he trained as a paratrooper in the Pararescue special forces outfit of the US Air Force. He soon turned his efforts to the more contemporary meaning of his name: "a fellow is often part of an elite group of learned people who work together as peers in the pursuit of knowledge or practice" [16]. However, also within the safe haven of academia Mike Fellows remains a fighter and a comrade in arms, fearless in his pursuit of knowledge and friendship alike.

Mike did his graduate studies in the 1980s at the University of California in San Diego, receiving a Master of Arts in Mathematics and a Ph.D. in Computer Science. In many Eastern religious traditions, one can attain Knowledge and Enlightenment only through a teacher already possessing these traits, with the Sanskrit word *parampara* denoting "the line of spiritual gurus in authentic succession of initiation; the chain of mystical power and authorized continuity, passed from guru to guru" [16]. A computer scientist will see here a recursive definition. While leaving the base case undefined, suffice it to say that Mike's ancestral line of supervisors contains some very famous names in mathematics, amongst them the Norwegians Sophus Lie, Axel Thue and Thoralf Skolem, via the graph theorist Øystein Ore, to the Americans Marshall Hall, Donald Knuth, and Mike's own supervisor Michael Fredman [13].

In the 1980s the pursuit of faster computing was tied to message-passing parallelism, which involved the mapping of a parallel computation to a parallel architecture and opened the way for some graph theory. Mike's thesis "Encoding graphs in graphs" was motivated by these issues. In general, you have a computation graph G and an architecture graph H and ask for a mapping of vertices $f : V(G) \rightarrow V(H)$ and a mapping of edges of G to paths in H so that $uv \in E(G)$

is mapped to a path between $f(u)$ and $f(v)$. The goal is a load-balanced mapping with low dilation and contention. A mathematician at heart, Mike's main interest seems to have been the very structured cases. The above mapping is called a *covering* of H by G if the edges of G are mapped to edges of H , i.e. paths of length one, in such a way that edges incident with a vertex v in G are mapped bijectively onto the edges incident with $f(v)$ in H . For connected H the mapping is in this case perfectly load balanced and the related computational question is called the H -COVER problem: given a graph G , does there exist a covering of H by G ? A paper by Mike with co-authors Abello and Stillwell [1] was the first to show existence of a graph H for which H -COVER is NP-complete: take H the graph on two adjacent vertices and a loop on each vertex. But there are also polynomial-time cases, e.g. only cycles of length a multiple of q will have a covering to the cycle of length q . For k -regular graphs we actually have a dichotomy: unless $P=NP$, H -cover for a k -regular graph H is polynomial-time solvable if and only if $k \leq 2$, see the survey [11]. However, a characterization of the graphs H defining a polynomial-time solvable H -COVER problem is still a wide open problem. What about taking H as a parameter? It is actually an open question if the parameterized viewpoint contributes anything in this case, since any graph H for which we know H -COVER to be FPT parameterized by H we also know it to be polynomial with H as part of the input.

The most famous open problem concerning graph coverings is probably Negami's conjecture from 1988 stating that a graph H has a planar cover (i.e. a finite planar graph G covering H) if and only if H embeds in the projective plane [14]. The 'if' direction is easy, and graphs embeddable in the projective plane are characterized by 32 connected forbidden minors. The main line of attack on the conjecture consider each of these forbidden minors to show that they do not have a planar cover. After many years of work a single graph remains, and the Negami conjecture has been reduced to showing that the graph $K_{1,2,2,2}$ does not have a planar cover, see the survey [12]. In an unpublished manuscript from 1988 [9] Mike made a related conjecture. An *emulation* is a mapping slightly less structured than a covering, requiring only that edges incident with each vertex v in G map surjectively (rather than requiring bijectivity) onto the edges incident with $f(v)$ in H . Mike showed that the property of having a planar emulator is preserved under taking minors and under $Y\Delta$ -transformations and conjectured that a graph has a planar emulator if and only if it has a planar cover. This conjecture has remained firmly tied to Negami's conjecture over the years. However, in a surprising turn of events (called a "breakthrough" by one of the main experts on Negami's conjecture [12]) Rieck and Yamashita showed very recently [15] that $K_{4,5}-4K_2$ and $K_{1,2,2,2}$ have planar emulators, and since the first of these graphs has been shown *not* to have a planar covering, this disproves Mike's bold conjecture. Mike seems to have been the first to introduce the concept of emulators, which has later been studied under the name of role assignments with applications in the theory of social behavior [6]. In that case H models roles and their relationships in a society, G represents relations between a group of

individuals, and the task is to assign roles to individuals so that each individual with a particular role has, among its neighbors, every role prescribed by H , and no other roles.

At a conference in New Zealand in 1990 Mike meets a local complexity theorist over a bottle of Villa Maria Cabernet Merlot and discover that they share many interests [5]. The resulting Downey-Fellows collaboration in parameterized complexity is still going strong and is our field's nearest equivalent of a Lennon-McCartney trademark. Being the new kid on the block, parameterized complexity has had to fight hard for its recognition, and Mike has been forced to put several of his talents into play, including his interests in theatre and education. In his Advice to Students he writes: "Story is central. Story is a bigger force than science. Everybody lives by stories. They are a primal force. In mathematics, we add formalism. We have equations that lead to solutions but story has its own logic. Find the story in what you are telling and presenting. This will help the listener meet you more than half-way" [10]. In every talk of Mike Fellows there is a clear storyline, and there are many of us who have been hooked on the parameterized viewpoint ever since first hearing about this suggested deal with the devil of intractability. In 1997 Mike spent a couple of weeks at the Department of Informatics at the University of Bergen in Norway and inspired us not only to do research in parameterized complexity, but also in education of children. Applying exercises taken from his book 'Computers Unplugged' (written with Tim Bell and Ian Witten) [2] we visited elementary schools to teach concepts like graph coloring. I vividly remember the local head teacher looking at the classroom of 11-year-olds merilly coloring the various graphs given to them on hand-outs and wondering aloud if this really had anything at all to do with mathematics or computers. Meanwhile, one of those kids produced a proof that a graph is 2-colorable if and only if it has no odd cycles.

When rumors started circulating in 1999 that Mike was getting married there were many of us who thought his days as a travelling mathematician were over. How wrong we were proven to be! Whereas the travelling mathematician par excellence is Paul Erdős, the travelling mathematical couple of the last decade is surely Mike and Fran. Frances Rosamond is Mike's life companion in all aspects, and shares his adventurous spirit and love for mathematics and education. Not only did they keep up the round-the-world weaving of the parameterized web by their own travelling, they also invited groups of young researchers for prolonged stays in their own home, and here I am stumped in my search for previous role models. In 2002 my then PhD student Christian Sloper was invited to work with Mike in Newcastle, Australia. Below is a report he recently sent me of his visit. Keep in mind that the Australian-based couple and their young visitors are a generation apart and that young mathematicians/computer scientists are not likely to prioritize household chores like cleaning, cooking or laundry.

They are both very inclusive, not excluding anyone. I remember knocking on their door the first day, but nobody opened, so I just went in and found them in the living room. I was very well received and was immediately served some food. The next day Mike would teach me surfing, which for

a bad swimmer like me was a scary experience. Not that he thought it a problem that I did not get to surf, he was rather more upset about how little I knew of world literature. There was one bedroom upstairs, two bedrooms downstairs for the guests, and a storage room for surfboards and wetsuits. Either Mike or Fran would cook, and the meals were always matched with some wine.

While I was there we were only two guests, me and David, but at other times there were more. The furnishing of the house had a rather spartan aspect to it, with a flipchart being one of the essential items. Mike really likes to stand by a board and explain, "the man needs an audience" as Fran said. So he drew figures and explained, and was open to our comments. He was responsive to suggestions. I later came to see that he often gave our ideas a lot more praise than they really deserved. He was good at viewing the little things that we came up with in a positive light. In this sense he was very encouraging.

We lived there for a few weeks, but one day when they arrived back from one of their camping tours up and down the coast it became clear to all that it was time for me and David to move on. On these camping tours the surfboards naturally went along and also a flipchart, so Mike could explain his ideas to Fran. Surfing is supposedly best done in the morning before the sun has warmed the air over land and turned the wind, so Mike surfed mostly in the morning. But even in the middle of discussions he would follow the beaches by webcam and now and then suddenly disappear when the wind conditions improved.

After I moved out I met with Mike almost every day at the University. How much we actually accomplished together is less clear, as I did not produce massive amounts of articles down there, and most of them were with Elena. When we did things together with Mike, he was always very clear that anybody who had participated should be a co-author, so that's very good. To illustrate how inclusive they were I recall they incorporated in some discussions also the wife of a colleague since she apparently was good at problem-solving, in particular crossword puzzles. To me this seemed really more an inconvenience than anything else as she did not know any computer science. I don't think she made it as co-author, though. (*by C.Sloper, translated from Norwegian by the author*)

Many young researchers can testify to Mike and Fran's inclusiveness. On later visits to our by now burgeoning algorithms research group in Bergen, Mike has always been accompanied by Fran, and the two of them have kept up a schedule that would be unimaginable to most. Not only in the amount of travelling from place to place, but also for their mode of work, where they invite collaboration from anyone, and could end up with five research meetings on the same day, with different groups of people. Lucky are all the young people who have benefited from these opportunities to work with the leading figure in the field. At least for our research group this collaboration with Mike has been indispensable.

Here is another quote from Mike's Advice to Students: "Some problems have important applications, while others have the potential to build theory. Some people are natively problem solvers with sharp tools and others are theory builders with a big picture view, although probably all are a bit of both." Safe to say that Mike is a lot of both! He is never afraid of delving deeply into a technical reduction. A prime example is the problem of computing the cliquewidth of a graph, a parameter first introduced in 1993 [4], with virtually everyone realizing it must be NP-complete, but aach, nobody able to show it. Cliquewidth is a slippery fish. For all that anybody knows it may even be that removing a single vertex may drop its value by half. In 2006, in a technical tour-de-force, by forcing the optimal cliquewidth expressions of a class of graphs into a more compliant linear structure, Mike, Fran, Udi Rotics and Stefan Szeider managed to hold the cliquewidth fish long enough to make an NP-completeness reduction [8]. As regards Mike and theory building, history speaks for itself. Possibly the notion of kernels will turn out to be the most lasting. At least they are the simplest to explain. Every practitioner knows that confronted with a difficult problem instance you first do some easy pre-processing to reduce the instance down to its hard kernel. How can we model this for an NP-hard problem? Could we ask for a reduction of each instance to a smaller one in polynomial-time? Alas no, that would imply $P=NP$. But surely there must be cases where we can reduce at least the big instances? Yes, indeed, and we are forced to introduce a parameter k besides input size n to define "big" as $n > f(k)$ for some function $f()$. So the parameter appears naturally for two reasons, one practical, since for many applications the instances come equipped with a small parameter, and the other also practical, as the only way to account for the pre-processing that we do in any case. The notion of polynomial-sized kernels has the added benefit of tying into classical complexity theory [3], and today it is clear that parameterized complexity is no longer the new kid on the block.

Congratulations Mike, may you continue weaving the web of mathematics and adventure for many new generations of young scientists!

References

1. Abello, J., Fellows, M., Stillwell, J.: On the Complexity and Combinatorics of Covering Finite Complexes. *Australasian Journal of Combinatorics* 4 (1991)
2. Bell, T., Fellows, M., Witten, I.: *Computer Science Unplugged: offline activities and games for all ages* (1996)
3. Bodlaender, H., Downey, R., Fellows, M., Hermelin, D.: On problems without polynomial kernels. *J. Comput. Syst. Sci.* 75, 8 (2009)
4. Courcelle, B., Engelfriet, J., Rozenberg, G.: Handle-rewriting hypergraph grammars. *J. Comput. System Sci.* 46, 218–270 (1993)
5. Downey, R., Fellows, M.: *Parameterized Complexity*. Springer (1999)
6. Everett, M., Borgatti, S.: Role colouring a graph. *Mathematical Social Sciences* 21(2), 183–188 (1991)
7. Fellows, M.: *Encoding Graphs in Graphs*. Ph.D. Dissertation, Univ. of California, San Diego (1985)

8. Fellows, M., Rosamond, F., Rotics, U., Szeider, S.: Clique-Width Minimization is NP-Hard. In: Proceedings of the 38th Annual Symposium on Theory of Computing, STOC 2006 (2006)
9. Fellows, M.: Planar Emulators and Planar Covers (1988) (manuscript)
10. Fellows, M.: Advice to Students,
<http://www.mrfellows.net/wordpress/advice-to-students-2/>
11. Fiala, J., Kratochvíl, J.: Locally constrained graph homomorphisms - structure, complexity, and applications. *Computer Science Review* 2, 97–111 (2008)
12. Hliněný, P.: 20 Years of Negami's Planar Cover Conjecture. *Graphs and Combinatorics* 26, 525–536 (2010)
13. The Mathematics Genealogy Project
14. Negami, S.: The Spherical Genus and Virtually Planar Graphs. *Discrete Math.* 70, 159–168 (1988)
15. Rieck, Y., Yamashita, Y.: Finite planar emulators for $K_{4,5}$ - $4K_2$ and $K_{1,2,2,2}$ and Fellows' Conjecture. *Eur. J. Comb.* 31, 903–907 (2010)
16. Wikipedia

Passion Plays: Melodramas about Mathematics

Frances Rosamond

School of Engineering and Information Technology
Charles Darwin University
Darwin, Northern Territory 0909 Australia
`frances.rosamond@cdu.edu.au`

Abstract. Most people don't know that Michael Fellows' efforts to introduce mathematics to the public went beyond *Computer Science Unplugged* and engaged a bold, new venue — inventive, inquisitive theatre. He wrote several plays, but this chapter will only give a brief description of the *Four Cowboy Melodramas of Mathematics*. The term “melodrama” refers to a dramatic work that exaggerates plot and characters in order to appeal to the emotions. Each play proves at least one mathematical theorem. The dramas have been played on stage only a few times.

1 Introduction

Mike began to consider how best to communicate the metaphors of mathematics through theatre. In notes about mathematics and theatre, Mike wrote, “What is mathematical science really all about? I believe it is about the unfolding of our collective abstract cognitive abilities, as part of our natural instinct to develop rich and expressive, as well as useful language. Mathematical science is therefore destined for the theatre, as it is powerfully and inherently metaphorical.”¹

In a 39 page unpublished manifesto on mathematical communication written about 1995, Mike described the potential of mathematical theatre for exploring the “cultural politics of curiosity” and “new cultural energies circulating in the world of mathematical science,” particularly attracting and engaging the intellectual interests of wider audiences. Mike began to experiment with how to communicate sophisticated technical information on stage. He thought of this as “content-driven” theatre.

In 1997, Mike was awarded a fellowship to the Centre for Studies in Religion and Society at the University of Victoria. It was the first given to anyone in computer science. The title of his application was *Religious Imagery, Mathematical Metaphor and Popular Participation in Science: An Exploration of Passion Plays*. His abstract points out that “The use of mathematical metaphor in religious contexts is striking and has a long and important history, including perhaps the roots of the dominant modern scientific world view (that the intellect is superior to the senses in discerning timeless truths) in the Pythagorean religion of mathematics”.

¹ In addition to the original play scripts and transparencies, I have had access to Mike's hand-written notes and papers to use for this chapter.

2 The Four Melodramas

Mike wrote four cowboy melodramas (think virtue tied to the railroad track by mustached villain). At a superficial level, the plays are almost vaudeville. They may be considered satires on standardized aspects of the school system. At a more serious level, they address issues in the politics of curiosity and are expected to be provocative. Each play dramatizes deep mathematics that has been part of Mike's professional research. Mathematics is feared by most people. Mike understood that putting mathematics on stage in any form would challenge the audience. He hoped to expose mathematics as a source of beauty, surprise and humility, and of fresh metaphorical insight into problems of every possible kind. He wrote:

Mathematics is at the heart of the scientific and technological Mega-Machine that we have created. Part of our task is to own this Machine, joyfully and not in terror. We must come to grips with issues such as expressed by Gödel in interpreting his own beautiful theorem:

“Either Mathematics is more than Man, or Man is more than a Machine.”

Todd Wareham suggested that the Cowboy Melodramas belonged to the genre of “passion plays,” in the sense of A. C. Cawley. Cawley described the primary concern of the medieval playwright as instruction to the audience through dramatic entertainment on the stage. Medieval morality stories were imbued with the folk-tale in order to include comic and human popular influences. Mike's love of literature and Heraclitus sense of culture shows free reign in the creation of poetic, fantastic and folk-tale elements in the plays.

The moralities in Mike's plays revolve around the ownership of science, and the opportunities for participation. Indifference of science professionals, and the tendency to treat knowledge as a ‘secret’, have contributed to the widespread alienation of the public from science. Mathematical science, which is the engine of modern life, which is more a culture of questions than answers, which consists of childlike curiosity, has been kept hidden from children.

3 *The Helen Keller of Arithmetic Story*

The first play is called *The Helen Keller of Arithmetic Story*. The second play is titled *Dragons*. Together they make a volume called *Visions of Joanna, A Cowgirl at the Edge of the Millennium*. (Helen Keller was an American author, political activist, and lecturer. She was the first deaf-blind person to earn a Bachelor of Arts degree.)

The mathematical theme of the *Helen Keller of Arithmetic Story* is the well-quasi-ordering of finite trees and Kruskal's Theorem. The Peano Arithmetic independent miniaturization and its associated fast-growing function plays a role in the showdown scene. Other topics making brief appearances include sorting networks, partial orders and Goldbach's conjecture.

The satire in the *Helen Keller of Arithmetic Story* is evident in the following description by University of Victoria, Canada Professor Maarten van Emden, from his 1998 diary.

The play only makes its appearance after an enactment of a summary of Fellows' personal experiences in being exposed to the conventional wisdom in Education Land. Choice snippets: Piaget having proved that children only become ready for Infinity at the age of 15 and a half, the experiences of Joanna, age 7 are shown to contradict this. The games Joanna thinks up include tying a rope to a table leg and producing single waves at a time, then standing waves. She plays a game with her dog in which perverts like computer scientists recognize a formal grammar.

Just as lecturers are advised to start with a transparency giving an outline, Fellows tells about a play being planned, gives the outline on a transparency in the traditional professorial style and starts explaining how the play goes, whereupon the play, well, goes. In between scenes he goes back to the colourful, clever transparencies, which were created by Lisa Whittle. The transparencies and a bit of costume help Mike, in these one-man shows, change quickly from one character to another.

An early scene relates the traumatic experience of two children. While they are in the charge of the resident clown of the shopping Mall, their parents are swallowed by the Mall. This is illustrated by a transparency showing a monstrous Mall. Overlays show exclamatory clouds in the vein of: "23% off, 312 days warranty!" against "18% off, 777 days warranty!!!" Obviously, Math is important. As a result, the children go blind and deaf to arithmetic.

In the story, an old hermit (a transparency of Erdős in cowboy garb), visits the orphan children who have tragically become blind and deaf to arithmetic. He describes the notion of topological embeddings of trees, of one tree including another inside of itself, with the same root, and instructs the children that they have "the job of growing until your personal tree has grown big enough and rich enough in structural form to include all the trees that you have loved and tragically lost...until those lost trees that you have loved, live again inside of you."

The Good Cowboy demands that the standardized math exam be offered in Braille. All the children pass. The widow's lands are saved, and the villain is booted.

In 1999, Geri Lorway and I helped Mike present *Helen Keller* to 400 teachers at the California Mathematics Teachers Association at Asilomar. Mike constructed a special sorting network by putting thick cord under the tape that defined the network (the Sorting Network is described in the chapter on Computer Science Unplugged). Teachers pretended to be the blind and deaf children, and we had them take off their shoes, close their eyes, and with their toes feel their way

along the network. Geri remembers being impressed that Mike spontaneously asked if anyone played piano, finding a volunteer and then just asking her to do some cowboy music at his cue.

After the presentation, several teachers came up to Mike crying. They were quite emotional, saying that the plays reached their deep inner feelings, confusions and brought back memories of early school days. The emotive power of the plays was very high, and likely was related to the Paranoid Theory (described in Section 3 of the chapter on *Computer Science Unplugged*).

Geri Lorway writes about the plays:

I believe it is a critical part of the whole that is not just Mike, but his work and the thing that is mathematics, the sciences and CS..... there is something so important in the deep emotional and if I can say spiritual impact that the idea of the passion plays brings to understanding why things like *CS Unplugged* are such phenomenal works.... the opportunity to bring the whole into one's understanding of learning, thinking, being human.

We must stop being such snobs about the *entireness* that feeds into genius.... the creative spirit is so intense it has to be allowed free reign and that is what those plays were.... they made some people catch their breath and say: My god there is way more to what I thought was the most horrible thing ever imagined: "mathematics"

4 *Dragons*

The second play in the *Visions of Joanna* series is called *Dragons*. The satire is darker, and begins with parents being burned at the stake. They had been subversively pleading for "thinking skills." Their daughter runs into the forest. She returns disguised as a baby dragon, to lead a children's education revolution.

The mathematics is a dream-story centered on the Circular Braid Theorem (CBT). Props for the CBT are three tangled pieces of heavy, large-diameter ship-mooring rope, colored red, green and blue, which a shepherd dumps from his duffel bag, signifying the confusions of his village. The green rope symbolizes the work of the villagers in the fields. The blue symbolizes their collective community, and the red rope symbolizes terrifying dragons of which everyone lives in fear. The story of the shepherd's vision (and the proof of the CBT) begins with a "lightning flash" staged by a camera bulb, together with yellow construction tape that flutters down from above the stage onto the tangle of ropes, with the initial tangle being the Boromean rings.

In addition to the proof of the CBT, mathematical content includes cameos about fractals and cryptographic one-way constructions to create computationally hard puzzles with which to paralyze the dragons.

The CBT proof involves marking positive and negative crossings of the yellow construction ribbon (the path of illumination) and each of the colored ropes.

To illustrate this, the shepherd describes wandering the hills as a young man, following old paths of confusion (a metaphor for the school system). He kneels down as if studying a large map, and makes a walking figure with his fingers that walks along the green rope, musing "The path of fields and work: day and night, day and night (As he talks, he marks with pieces of silver tape the positive and negative crossings of the green rope with the yellow ribbon)...and remembering each encounter with the vision.

He continues on the blue rope. "Remembering each time the path of our people met the illumination. The pride, the shame, the pride, the shame." (Now marking the positive and negative crossings of the blue rope with the yellow ribbon).

He does the same with the red rope path of the dragons. "What are these dragons? How do they live? The burning curiosity, the fear, the curiosity, the fear...!"

In the next step of the proof, the shepherd takes a long thin multicoloured cord from his bag, and uses it to weave over and under through the tangle according to the markings. As he weaves, the shepherd uses the words "over" and "under" to describe the possibility of a life that has overcome the confusion in the village.

"I imagined a life that would follow the path of the illumination, yet a life not just knowing what is, uncaring -- but a life of value! Under the shame, and over the pride, under the darkness, and over the light, under the fear, and over the curiosity..."

"A life that could separate the bright from the dark. A spirit that could soar above in the brightness, and delve deeply to rest in the darkness...."

"A life that could separate the pride from the shame. A spirit that could reach beneath the shames and know them, and overcome the shallow prides...."

"A life that could separate the fear from the curiosity. A spirit that could reach down beneath the fears to touch their shivering roots, and reach above the curiosities towards enlightenment...."

"And then I reached out my hand to see where that life, if I chose to live it, would take me..."

While speaking, the shepherd has been putting his hand along the multi-coloured path, over and under the big ropes and creating the circular braid. The story and the proof reach a crescendo, with mathematics (the CBT) empowering a new way of seeing reality.

"And this is what I have found! Do you see how the circles of our lives and work in the fields - our peoples - and the dragons - are all woven in hidden parallels together in the same circle of being - do you see? This is our life. There is no confusion."

There are other mathematical elements in *Dragons*, such as graph three-coloring, enacted during a children's revolution with colors indicated by body positions: arms raised overhead, arms stretched out from the sides, squatting. As with all the plays, Mike is the only actor.

4.1 *Bob, Cowboy Mathematician of the Yukon*

In the third play, *Bob, Cowboy Mathematician of the Yukon*, Mike used Parameterized Complexity and religion to address big issues of ownership and participation in mathematics. The first scene shows Cowboy Bob explaining computational complexity to interested children in a local school. Towns-people are horrified when they become aware of Bob's religious metaphor, that parameterized complexity "exploits thin zones of parametric viability" – small ranges of parameters, for which we can work out a "deal with the Devil." They begin to mumble, "Devil-worshipper." When Bob makes an isomorphic relation to biological life, which also inhabits a largely hostile universe, the towns-people become increasingly intolerant (think of America's "evolution vs creationism" controversy). A third interlocking theme becomes the gradual accretion of layers of understanding and meaning, the stories we attach to the rock of fact. Bob calls this "The Talmud of Story Science."

The meanings that various people have of mathematics come out when they answer Cowboy Bob's question, "Would Space Aliens have the same theorems we do?" (This question was raised in one of Mike's classes at UVIC. By "Space Aliens" is meant beings with enough mathematical sophistication to design and fly spacecraft through the universe. Mike was taken aback that there were a variety of answers – even when the question was rephrased to be more specific, "Would Space Aliens have the same theorems about the prime numbers?" He began to carry the question around to scientific colleagues, who also had a variety of answers.)

The last scene finds Cowboy Bob in a saloon talking to his son. "Each of us is a computational creature, faced with the fundamental problem of being finite. The most important parameter of all is the individual." Bob concludes that when the Space Aliens arrive, they are going to want to talk to scientists, the people who are "interested in the Devil". A crowd has gathered outside. They are shouting: "Den of iniquity! Send out the devil worshipper!" The hymn Onward Christian Soldiers is heard.

Bob, Cowboy Mathematician of the Yukon, and the two plays in *Visions of Joanna, A Cowgirl at the Edge of the Millennium*, were a response to what Mike saw as a school math curriculum that had been stripped down to, as Mike put it, "18th century shopkeeper arithmetic – grocery arithmetic." He pointed out that while there is nothing wrong with arithmetic, mathematics is much, much more — the essential engine of modern science. Teaching such a narrow view of mathematics is a tragedy and a crime against children.

4.2 *Wagon Train to Infinity*

The final melodrama, *Wagon Train to Infinity*, is the only one of the four plays that is not a satire on schools. This play focuses on what it means to be a spiritual individual in modern science and society. One might say that Mike's personal moral imperative is compassion and a creative attitude toward life. Sometimes Mike says, "There is a lot to be said about Attitude."

Wagon Train to Infinity stars the Lovasz Local Lemma, some appearances of Kestens Theorem on percolation in the plane, and results on graphical evolution. The wagon train is going to Infinity, which represents mathematical literacy for everyone (It's like Oregon, with lands that go on and on and on). The reason for going is that it is fun there. It gives a useful perspective. Because we get the odd bit of comfort there. Because it is a fair place. Along the way, the Wagon Train is attacked by the Intuits (e.g., fundamentalists).

A Wagon Train is not just about where one is going, but what one is leaving behind. In this story, they are leaving behind artists ignorant of mathematics, cultures with no curiosity, and teachers who give you a "C" because they don't like your opinions.

The Preacher character equates Infinity with the New Testament, and a yearning to "get beyond the finite numbers" on the way. The finite numbers to go beyond are those of modern atrocities, such as the number of deaths in Argentina, or the number of minutes before unconsciousness from electroshock torture. These are all intoned as if from the *Book of Numbers*, in Old Testament style. The Preacher's sermons correspond to basic theorems about Cantor diagonalization, and he enacts the ghost dance of the uncountable.

4.3 Diagonalization

The notion of diagonalization is a powerful metaphor for Mike. In a festschrift chapter in honor of the humanistic mathematics educator Stephen I. Brown, Mike used diagonalization to describe Christianity (partly as a means of describing Steve's work). Each person who accepted the Invitation to take up their (individual) cross and follow, contributed to the power of the invitation. In the extended metaphor, each person was a "row" and the Invitation the diagonal. Moving beyond ancient religious human sacrifice cults, the metaphor witnesses a new kind of civilization formed when a creative response comes from each individual. All the individual sacrifices became embodied in the "new" sacrifice. Each individual human being is a creative collaborator with God.

Mike has even used diagonalization as a metaphor for his favourite sport, *surfing*! In surfing, the analogue of one of Cantor's rows is: "heading directly into the beach." Instead of taking that ride, diagonalizing "tweaks" it, and we move a bit sideways. Now from our new spot we could still go directly to the shore. We don't take that one either. We go on a diagonal. We enjoy going on a diagonal to explore the complexities of real currents.

The steadfast confidence that Michael Fellows has in human curiosity, and his sense of justice that the natural curiosity of children not be impeded or thwarted, drive his plays forward. His belief that mathematics is a metaphor on multiple layers makes his theater relevant to everyone in all phases of their lives.²

References

1. Rosamond, F.: On-line and off-line computer games and mathematical sciences popularization. In: Rosamond, F., Copes, L. (eds.) *Educational Transformations: The Influences of Stephen I. Brown*, Authorhouse, Bloomington, Indiana, pp. 407–426 (2006)
2. Fellows, M.: A short meditation on Steve Brown’s main pedagogical idea, as related by Frances Rosamond. In: Rosamond, F., Copes, L. (eds.) *Educational Transformations: The Influences of Stephen I. Brown*, Authorhouse, Bloomington, Indiana, pp. 400–406 (2006)
3. Cawley, A.C. (ed.): *Everyman and Medieval Miracle Plays*. Introduction by J.M. Dent 1974, Everyman 1956, Orion Publishing group, Orion House 5, Upper St. Martin’s Lane, London WC2H 9EA, ISBN 046087280X

² On a personal note, although Mike and I had corresponded about mathematics popularization activities, we had not met until he invited me to come up from San Diego to see the plays. They were being presented at the 1998 Fringe Theatre Festival in Victoria, Canada. By the end of that weekend, we had decided to get married.

We have enjoyed the visits of many researchers to our home in Australia. Quite often, Mike receives an email saying something like, “You don’t know me, but I have become interested in Parameterized Complexity, and I have this question...” Mike’s response is to arrange for the person to come visit, and to set up the flip-charts. We have a couple of traditions, in addition to sharing *Unplugged*, the plays, and “Mr. Opinion” (*The Guide to Modern World Literature*, a reference book by Martin Seymour-Smith.) One is for students to cook a meal from their native country, and this has resulted in charming phone calls home asking Mom for recipes and instructions. Another is teaching the visitor to surf – but we haven’t asked them to diagonalize! Yet, many do so naturally.

Part II
Surveys

A Basic Parameterized Complexity Primer

Rod Downey*

School of Mathematics, Statistics and Operations Research
Victoria University
P.O. Box 600, Wellington, New Zealand
rod.downey@vuw.ac.nz

Abstract. This article was prepared for Mike Fellows Festschrift for his 60th Birthday. Since many of the contributed articles revolve around the concept of parameterized complexity, it seems reasonable to give the reader a (short) primer to this area. It is not intended as a complete survey of this very broad area in its current state; rather it is intended to give a flavour of the techniques used and the directions taken. Whilst not doing the area justice, the basics of the techniques for proving tractability, establishing hardness, and the philosophy are given. The basics from this paper will be amplified by many other articles in this Festschrift. Much fuller accounts can be found in the books Downey-Fellows [DF98, DFta], Niedermeier [Nie06], Flum-Grohe [FG06], the two issues of the *Computer Journal* [DFL08] and the recent survey Downey-Thilikos [DTH11].

1 Introduction

1.1 The Idea

The story of classical complexity, as witnessed by the classic cartoons in the beginning of Garey and Johnson's book [GJ79], begins with some problem we wish to find an efficient algorithm for. Now, what do we mean by efficient? It seems a reasonable idea to idealize the notion of being efficient by being in *polynomial time*. Having done this, we discover that the only algorithm we have for the given problem is to try all possibilities and this takes $\Omega(2^n)$ for instances of size n . What we would like is to *prove* that there is no algorithm running in feasible time. Using our idealization that feasible=polynomial, this equates to showing that there is no algorithm running in polynomial time.

Suppose that we succeed in showing that there is no polynomial time algorithm. This would mean to us is that we would (i) need to try some other method to solve the problem such as some kind of approximate solution because (ii) we could give up on showing that there was a polynomial time algorithm.

The story continues with the following rhetoric. In spite of the efforts of a number of researchers, for many problems whose best solution known was complete search, there was no *proof* that the problem is not in polynomial time

* Research supported by the Marsden Fund of New Zealand. Dedicated to my old friend Mike on the occasion of his 60th Birthday.

found. It was the key realization of Cook, Levin and crucially Karp [Ka72] that many of these problems could be shown to be polynomial-time reducible to each other, and to the problem of acceptance for a polynomial time nondeterministic Turing machine. That is, they are NP-complete. This means that we have a *practical* “proof” of hardness in that if any of the problems were in polynomial time, all would be; and secondly showing them to be in polynomial time would show that acceptance for a polynomial time nondeterministic Turing machine would be also. The philosophical argument is that a nondeterministic Turing machine is such an opaque object, without any obvious algebraic structure, that it seems impossible to see if it has an accepting path without trying all of them. That’s the philosophy anyway.

The methodology above seems fine as a *first foray* into *feasible* computation. However, for practical computation, it seems that we ought to refine the analysis to make it more *fine grained*. Firstly, when we show that something is NP-complete or worse, what we are focusing on is the worst case behaviour. Second, the analysis takes the input as being measured by its *size* alone. You can ask yourself the question: when in real life do we know nothing else about a problem than its *size*? The answer is *never*. For instance, the problem is planar, tree-like, has many parameters bounded, etc. The idea behind parameterized complexity is to try to exploit the *structure* of the input to get some practical tractability. That is, we try to understand what aspect of the problem is to blame for the combinatorial explosion which occurs. If this parameter can be controlled then we would have achieved practical tractability.

Anybody working in software engineering will know that it is important to design tools specific to the type of problem at hand. Suppose that you are concerned with relational databases. Typically the database is huge, and the queries are relatively small. Moreover, “real life” queries are queries *people* actually ask. Hence, such queries tend to be also of low logical complexity (see, for example Papadimitriou and Yannakakis [PY97], which posits parameterized complexity as the correct complexity for such analyses). Furthermore, in areas like computational biology, the number 4 is typical, and structure of things like DNA is *far* from random. The *main idea* of parameterized complexity is to design a paradigm that will address complexity issues in the situation where we know in advance that certain parameters will be likely bounded and this might significantly affect the complexity. Thus in the database example, an algorithm that works very efficiently for small formulas with low logical depth might well be perfectly acceptable in practice.

Thus, parameterized complexity is a refined complexity analysis, driven by the idea that in real life data is often given to us naturally with an underlying structure which we might profitably exploit. The idea is not to replace polynomial time as the *underlying* paradigm of feasibility, but to provide a set of tools that refine this concept, allowing some exponential aspect in the running times by allowing us either to use the given structure of the input to arrive at feasibility, or develop some relevant hardness theory to show that the kind of structure is not useful for this approach.

As I remarked in [Do03], “This simple idea is pretty obvious once you think about it. For example, when we teach a first course in automata theory we show the students that regular language acceptance is in linear time. But this is really not quite true: it is only true *if* the language is presented to us as, say, a regular expression, whereas it could be a language presented as the output of a Turing machine, in which case acceptance is *undecidable*. The *point* is that we only really care about regular languages when they are given to us in a structured way, namely via regular expressions.”

1.2 Some Definitions

I will now discuss the standard examples which we use for the theory. As I discuss in the companion paper [Do12], Mike Fellows and my early work had the three problems VERTEX COVER, DOMINATING SET, INDEPENDENT SET in our hearts.

For a graph G a vertex cover is where vertices cover edges: that is $C = \{v_1, \dots, v_k\}$ is a vertex cover iff for each $e \in E(G)$, there is a $v_i \in C$ such that $v_i \in e$. They should recall that a dominating set is where vertices cover vertices: $D = \{v_1, \dots, v_k\}$ is a dominating set iff for all $v \in V(G)$, either $v \in D$ or there is an $e \in E(G)$ such that $e = \langle v_i, v \rangle$ for some $v_i \in D$. Finally an independent set is a collection of vertices no pair of which are connected. Of course, these are some of the basic NP -complete problems identified by Karp [Ka72].

As in [Do03], and earlier [DF98] and [DFS98], I will motivate the definitions by looking at a problem in computational biology. As discussed in [Do12] in this volume, and as seen by [GGKS95, KST94, St00, DFS98, BDFHW95] computational biology has been interacting with parameterized complexity from the beginning, and this interaction has continued with throughout, with the work Langston and his group (who have contracts throughout the world to analyse biological data, and use VERTEX COVER and other FPT techniques routinely), of Niedermeier and his group, and others. This volume describes many of the applications to computational biology in Stege [St12]. Suppose we had a conflict graph of some data from this area. Because of the nature of the data we know that it is likely the conflicts are at most about 50 or so, but the data set is large, maybe 10^{12} points. We wish to eliminate the conflicts, by identifying those 50 or fewer points. Let’s examine the problem depending on whether the identification turns out to be a dominating set problem or a vertex cover problem.

DOMINATING SET. Essentially the only known algorithm for this problem is to try all possibilities. Since we are looking at subsets of size 50 or less then we will need to examine all $(10^{12})^{50}$ many possibilities. Of course this is completely impossible.

VERTEX COVER. There is now an algorithm running in time $O(1.2738^k + kn)$ ([CKX10]) for determining if an G has a vertex cover of size k . This and structurally similar algorithms has been implemented and is practical for n of unlimited practical size and k large. The relevant k has been increasing all the time, evolving from about 400 in [CDRST03], to Langston’s team [LPSSV08, ELRW11] who now routinely solve instances on graphs with millions of nodes and vertex covers in the thousands. Moreover, this last work is on actual biological data.

As well as using bounded branching (and parallelization [ALSS06]), the method used for this algorithm for VERTEX COVER is called *kernelization* and is based on reduction rules¹, which tend to be easy to implement and perform often much better than anticipated in practice. We will discuss this method in detail soon. The following table from Downey-Fellows [DF98] exhibits the difference between the parameter k being part of the exponent like DOMINATING SET or as part of the constant like VERTEX COVER. This table compares of a running time of $\Omega(n^k)$ vs $2^k n$.

Table 1. The Ratio $\frac{n^{k+1}}{2^k n}$ for Various Values of n and k

	$n = 50$	$n = 100$	$n = 150$
$k = 2$	625	2,500	5,625
$k = 3$	15,625	125,000	421,875
$k = 5$	390,625	6,250,000	31,640,625
$k = 10$	1.9×10^{12}	9.8×10^{14}	3.7×10^{16}
$k = 20$	1.8×10^{26}	9.5×10^{31}	2.1×10^{35}

In classical complexity a decision problem is specified by two items of information:

- (1) The input to the problem.
- (2) The question to be answered.

In parameterized complexity there are three parts of a problem specification:

- (1) The input to the problem.
- (2) The aspects of the input that constitute the parameter.
- (3) The question.

Thus *one* parameterized version of VERTEX COVER is the following:

VERTEX COVER

Instance: A graph $G = (V, E)$.

Parameter: A positive integer k .

Question: Does G have a vertex cover of size $\leq k$?

We could, for instance, parameterize the problem in other ways. For example, we could parameterize by some width metric, some other shape of the graph, planarity etc. Any of these would enable us to seek hidden tractability in the problem at hand.

For a formal definition, for simplicity I will stick to the *strongly uniform* definition of being fixed-parameter tractable. There are other definitions of less importance in practice, and I refer the reader to [DF98] or [FG06] for more details.

A *parameterized language* is $L \subseteq \Sigma^* \times \Sigma^*$ where we refer to the second coordinate as the *parameter*. It does no harm to think of $L \subseteq \Sigma^* \times \mathbb{N}$. Flum and

¹ There are other FPT methods based around reduction rules such as Leizhen Cai [LeC96] and Khot and Raman [KR02], which work on certain hereditary properties.

Grohe have an alternative formulation where the second coordinate is a function $\kappa : \Sigma^* \rightarrow \Sigma^*$, but I prefer to keep the second parameter as a string or number.

Definition 1. A parameterized language L is (strongly) fixed parameter tractable (FPT), iff there is a computable function f , a constant c , and a (deterministic) algorithm M such that for all x, k ,

$$\langle x, k \rangle \in L \text{ iff } M(x, k) \text{ accepts,}$$

and the running time of $M(x, k)$ is $\leq f(k)|x|^c$.

It is not difficult to show that the multiplicative constant in the definition can be replaced by an additive one, so that $L \in \text{FPT}$ iff L can be accepted by a machine in time $O(|x|^c) + f(k)$ for some computable f . In the case of VERTEX COVER we have $f(k) = 1.2738^k$, and the O is 2. One nice notation useful here is the O^* notation which ignores the polynomial part be it additive or multiplicative and is only concerned with the exponential part. The algorithm would be said to be $O^*(2^k)$. The table on the web site

<http://fpt.wikidot.com/fpt-races>

lists 35 (at the time of writing) basic problems which are fixed parameter tractable with (mostly) practical algorithms, and for which there are current “races” for algorithms with the best run times.

Now you might (in some cases validly) complain about the presence of an arbitrarily bad computable function f . Could this not be like, for example Ackermann’s function? This is a true enough complaint, *but* the argument also applies to *polynomial time*. Could not polynomial time allow for running times like $n^{30,000,000}$? As noted by Edmonds [Ed65], the practical algorithm builder’s answer tends to be that “in real life situations, polynomial time algorithms tend to have small exponents and small constants.” That certainly was true in 1965, but as we will see this is no longer true. The same heuristic applies here. By and large, for most practical problems, at least until recently, the $f(k)$ ’s tended to be manageable and the exponents reasonable.

In fact, an important offshoot of parameterized complexity theory is that it does (sometimes) provide tools to show that *bad constants* or *bad exponents* for problems with algorithms *running in polynomial time* cannot be eliminated, modulo some reasonable complexity assumption. As articulated by Alekhovich and Razborov [AR01] who were considering lower bounds for the automatizability² of resolution and tree-like resolution, what they needed was a complexity theory *sensitive to the structure of polynomial time*. We emphasize that exploring

² Alekhovich and Razborov studied proof systems P called in [BPR01] *automatizable* meaning that there is a deterministic algorithm A which, when give a tautology τ returns its shortest proof in time polynomial in the size of the shortest P -proof of τ . They proved neither resolution nor tree-like resolution is automatizable unless $W[P]$ is randomized FPT by a randomized algorithm with one-sided error, where $W[P]$ is a class we will meet in Section 2

feasible computation requires something like parameterized complexity as it is a theory giving hardness *within polynomial time*. More on this in Section 2.

One of the key features of the theory is a wide variety of associated techniques for proving parametric tractability. We will discuss them in Section 3, but before we do so, let's examine the associated hardness theory.

2 Parametric Intractability

Since we are woefully bad at *proving* problems to be not in polynomial time, we have invented a hardness theory, as mentioned in Section 1, based on the assumption that certain canonical problems are not in polynomial time. The two key ingredients of a hardness theory are (i) a notion of hardness and (ii) a notion of “problem A could be solved efficiently if we could solve problem B ”; that is a notion of reducibility.

In the classic theory of NP completeness (i) is achieved by the following:

NONDETERMINISTIC TURING MACHINE ACCEPTANCE

Input: A nondeterministic Turing Machine M and a number e .

Question: Does M have an accepting computation in $\leq |M|^e$ steps?

The Cook-Levin argument is that a Turing machine is such an opaque object that it seems that there would be no way to decide if M accepts, without essentially trying the paths. If we accept this thesis, then we probably should accept that the following problem is not $O(|M|^c)$ for any fixed c and is probably $\Omega(|M|^k)$ since again our intuition would be that all paths would need to be tried:

SHORT NONDETERMINISTIC TURING MACHINE ACCEPTANCE

Input: A nondeterministic Turing Machine M

Parameter: A number k .

Question: Does M have an accepting computation in $\leq k$ steps?

So here is a notion of hardness. Personally I would find it difficult to believe that NP is not P, but that SHORT NONDETERMINISTIC TURING MACHINE ACCEPTANCE could be in FPT, for example, solved in $O(|M|^3)$ for any path length k . In fact, as we will soon see, SHORT NONDETERMINISTIC TURING MACHINE ACCEPTANCE not in FPT is closely related to the statement n -variable 3SAT not being solvable in subexponential time.

Thus to show DOMINATING SET is likely not FPT could be achieved by showing that *if we could solve it in time $O(n^c)$ by for each fixed k , then we could have a $O(n^c)$ for SHORT NONDETERMINISTIC TURING MACHINE ACCEPTANCE*. Our principal working definition for parameterized reductions is the following.

Definition 2. *Let L, L' be two parameterized languages. We say that $L \leq_{fpt} L'$ iff there is an algorithm M , a computable function f and a constant c , such that*

$$M : \langle G, k \rangle \mapsto \langle G', k' \rangle,$$

so that

- (i) $M(\langle G, k \rangle)$ runs in time $\leq g(k)|G|^c$.
- (ii) $k' \leq f(k)$.
- (iii) $\langle G, k \rangle \in L$ iff $\langle G', k' \rangle \in L'$.

A simple example of a parametric reduction is from k -CLIQUE to k -INDEPENDENT SET, where the standard reduction is parametric (a situation not common). The following is a consequence of Cai, Chen, Downey and Fellows [CCDF96], and Downey and Fellows [DF95b]; as I discuss in [Do12], elsewhere in this volume.

Theorem 1. *The following are hard for SHORT NONDETERMINISTIC TURING MACHINE ACCEPTANCE: INDEPENDENT SET, DOMINATING SET.*

Following Karp [Ka72], and then four decades of work, we know that thousands of problems are all NP-complete. They are all reducible to one another and hence seem to have the same classical complexity. On the other hand, with parameterized complexity, we have theory which separates VERTEX COVER from DOMINATING SET and INDEPENDENT SET. With such refined reducibilities, it seems highly unlikely that the hardness classes would coalesce into a single class like NP-complete. And indeed we think that this is the case. We have seen in the theorem above that SHORT NONDETERMINISTIC TURING MACHINE ACCEPTANCE \equiv_{fpt} INDEPENDENT SET. However, we *do not* think that DOMINATING SET \leq_{fpt} INDEPENDENT SET.

A standard parameterized version of the satisfiability problem of Cook-Levin is the following. (Other parameterized versions are discussed in the article by Chen and Flum [CF12] in this volume.)

WEIGHTED CNF SAT

Input: A CNF formula X .

Parameter: A number k .

Question: Does X have a true assignment of weight k (here the weight is the number of variables set to true)?

Similarly, we can define WEIGHTED 3 CNF SAT where the clauses have only 3 variables. Classically, using a padding argument, we know that CNF SAT \equiv_m^P 3 CNF SAT. Recall that to do this for a clause of the form $\{q_1, \dots, q_k\}$ we add extra variables z_j and turn the clause into several as per: $\{q_1, q_2, z_1\}$, $\{\bar{z}_1, q_3, z_2\}$, etc.

Now this is definitely *not* a parametric reduction from WEIGHTED CNF SAT to WEIGHTED 3 CNF SAT because a weight k assignment could go to any other weight assignment for the corresponding instance of 3 CNF SAT.

Now, early on, as I mention in [Do12], Fellows and I came to the belief that there is *no parametric reduction at all* from WEIGHTED CNF SAT to WEIGHTED 3 CNF SAT. Fellows and I proved that DOMINATING SET \equiv_{fpt} WEIGHTED CNF SAT. Extending this reasoning further, we can view WEIGHTED CNF SAT as a formula that is a product of sums. We can similarly define WEIGHTED t -POS SAT as the weighted satisfiability problem for a formula X in product of sums of product of sums... with t alternations. Fellows and I then defined WEIGHTED SAT if we have no restriction on the formula. Downey and Fellows [DF95a] called the

collection of parameterized languages FPT-equivalent to WEIGHTED 3 CNF SAT $W[1]$, the collection of languages FPT-equivalent to WEIGHTED CNF SAT $W[2]$, the collection of languages FPT-equivalent to WEIGHTED t -POS SAT $W[t]$, and the collection of languages FPT-equivalent to WEIGHTED SAT $W[SAT]$. There are some other classes $W[P]$, the weighted circuit satisfiability class, and XP which has as its defining problem the class whose k -th slice is complete for $DTIME(n^k)$, this being provably distinct from FPT and akin to exponential time. This gave the W -hierarchy below

$$W[1] \subseteq W[2] \subseteq W[3] \dots W[SAT] \subseteq W[P] \subseteq XP.$$

There are many, many problems hard for $W[1]$ and complete at many levels of this hierarchy. I won't list them here, but examples can be found in this volume, and in the papers and books listed above. The basic papers are [DF92a, DF92b, DF93, ADF95], and there we define the basic W -classes and essay the completeness programme.

The reader might ask about parameterizing *space*. This issue was addressed by Abrahamson, Downey and Fellows [ADF93, ADF95], where the complexity of k -move games was addressed. (It could be argued that a proper treatment of space is yet to be done.) The complexity of k -move games was addressed by extending the hierarchy above using ideas of *alternation* of parameterized quantifiers, giving a hierarchy called the AW -hierarchy. Again we refer to the books and survey articles.

There are also other hierarchies based on other ideas of *logical depth*. One important hierarchy of this kind was found by Flum and Grohe is the A -hierarchy which is also based on alternation like the AW -hierarchy but works differently. For a class Φ of formulae, we can define the following parameterized problem.

p -MC(Φ)

Instance: A structure \mathcal{A} and a formula $\varphi \in \Phi$.

Parameter: $|\varphi|$.

Question: Decide if $\phi(\mathcal{A}) \neq \emptyset$, where this denotes the evaluation of ϕ in \mathcal{A} .

Flum and Grohe define

$$A[t] = [p\text{-MC}(\Sigma_t)]^{\text{FPT}}.$$

For instance, for $k \geq 1$, k -CLIQUE can be defined by

$$\text{clique}_k = \exists x_1, \dots, x_k \left(\bigwedge_{1 \leq i < j \leq k} x_i \neq x_j \wedge \bigwedge_{1 \leq i < j \leq k} E x_i x_j \right)$$

in the language of graphs³, and the interpretation of the formula in a graph G would be that G has a clique of size k . Thus the mapping $(G, k) \mapsto (G, \text{clique}_k)$ is a fixed parameter reduction showing that parameterized CLIQUE is in $A[1]$. Flum and Grohe populate various levels of the A -hierarchy and show the following.

³ For narrative flow, I will assume that the reader is familiar with logic, but a more precise discussion will be given in Section 4.5.

Theorem 2 (Flum and Grohe [FG02a, FG04]). *The following hold:*

- (i) $A[1] = W[1]$.
- (ii) $A[t] \subseteq W[t]$.

Clearly $A[t] \subseteq XP$, but no other containment with respect to other classes of the W-hierarchy is known. It is conjectured by Flum and Grohe that no other containments than those given exist. This conjecture is not apparently related to any other conjecture. One other important hierarchy, called the M -hierarchy will be discussed later, but in any case it is completely evident that the fine-grained nature of the notion of parametric complexity will lead to significant structure within polynomial time. There is still a great deal to do here.

2.1 Connection with PTAS's

The reader may note that parameterized complexity is addressing intractability *within polynomial time*. In this vein, the parameterized framework can be used to demonstrate that many classical problems that admit a PTAS do not, in fact, admit any PTAS with a practical running time, unless $W[1] = FPT$. The idea here is that if a PTAS has a running time such as $O(n^{\frac{1}{\epsilon}})$, where ϵ is the error ratio, then the PTAS is unlikely to be useful. For example if $\epsilon = 0.1$ then the running time is already n to the 10th power for an error of 10%. Here is a table from Downey [Do03]

- Arora [Ar96] gave a $O(n^{\frac{3000}{\epsilon}})$ PTAS for EUCLIDEAN TSP
- Chekuri and Khanna [CK00] gave a $O(n^{12(\log(1/\epsilon)/\epsilon^8)})$ PTAS for MULTIPLE KNAPSACK
- Shamir and Tsur [ST98] gave a $O(n^{2^{2^{\frac{1}{\epsilon}} - 1}})$ PTAS for MAXIMUM SUBFOREST
- Chen and Miranda [CM99] gave a $O(n^{(3mm!)^{\frac{m}{\epsilon} + 1}})$ PTAS for GENERAL MULTIPROCESSOR JOB SCHEDULING
- Erlebach *et al.* [EJS01] gave a $O(n^{\frac{4}{\pi}(\frac{1}{\epsilon^2} + 1)^2(\frac{1}{\epsilon^2} + 2)^2})$ PTAS for MAXIMUM INDEPENDENT SET for geometric graphs.

Table 2 below calculates some running times for these PTAS's with a 20% error.

Table 2. The Running Times for Some Recent PTAS's with 20% Error

Reference	Running Time for a 20% Error
Arora [Ar96]	$O(n^{15000})$
Chekuri and Khanna [CK00]	$O(n^{9,375,000})$
Shamir and Tsur [ST98]	$O(n^{958,267,391})$
Chen and Miranda [CM99]	$> O(n^{10^{60}})$ (4 Processors)
Erlebach <i>et al.</i> [EJS01]	$O(n^{523,804})$

In Downey [Do03], I argue as follows.

“By anyone’s measure, a running time of $n^{500,000}$ is bad and $n^{9,000,000}$ is even worse. The optimist would argue that these examples are important in that they prove that PTAS’s exist, and are but a first foray. The optimist would also argue that with more effort and better combinatorics, we will be able to come up with some $n \log n$ PTAS for the problems. For example, Arora [Ar97] also came up with another PTAS for EUCLIDEAN TSP, but this time it was nearly linear and practical.

But this situation is akin to P vs NP . Why not argue that some exponential algorithm is just the first one and with more effort and better combinatorics we will find a feasible algorithm for SATISFIABILITY? What if a lot of effort is spent in trying to find a practical PTAS’s without success? As with P vs NP , what is desired is either an efficient⁴ PTAS (EPTAS), or a proof that no such PTAS exists⁵. A primary use of NP -completeness is to give compelling evidence that many problems are unlikely to have better than exponential algorithms generated by complete search.”

To use the hardness theory to eliminate the possibility of feasible PTAS’s, what we could do is regard $\frac{1}{\epsilon}$ as a parameter and show that the problem is $W[1]$ -hard with respect to that parameterization. In that case *there would likely be no method of removing the $\frac{1}{\epsilon}$ from the exponent in the running time and hence no efficient PTAS*, a method first used by Bazgan [Baz95]. For many more details of the method we refer the reader to the surveys [Do03, DTH11].

It was an insight of Cai and Juedes that tight lower bounds for approximation and parameterized complexity are intimately related; and indeed, are also related to classical questions about NP and subexponential time. In particular, Cai et. al. [CFJR07] who showed that the method of using planar formulae tends to give PTAS’s that are never practical. The exact calibration of PTAS’s and parameterized complexity comes through yet another hierarchy called the M -hierarchy.

The base level of the hierarchy is the problem $M[1]$ defined by the core problem below.

Instance: A CNF circuit C (or, equivalently, a CNF formula) of size $k \log n$, with n in unary.

Parameter: A positive integer k .

Question: Is C satisfiable?

That is, we are parameterizing the *size* of the problem rather than some aspect of the problem. The idea naturally extends to higher levels for that, for example,

⁴ An *Efficient Polynomial-Time Approximation Scheme (EPTAS)* is an $(1 + \epsilon)$ -approximation algorithm that runs in $f(1/\epsilon) \cdot n^{O(1)}$ steps. If, additionally, f is a polynomial function then we say that we have a *Fully Polynomial-Time Approximation Scheme (FPTAS)*.

⁵ The same issue can also be raised if we consider FPTAS’s instead of EPTAS’s.

$M[2]$ would be a product of sums of product formula of size $k \log n$ and we are asking whether it is satisfiable. The basic result is that $\text{FPT} \subseteq M[1] \subseteq W[1]$. The hypothesis $\text{FPT} \neq M[1]$ is *equivalent to* a classical conjecture called the *exponential time hypothesis*, ETH. This hypothesis is due to Impagliazzo, Paturi and Zane [IPZ01] and asserts that n -variable 3SAT cannot be solved in “subexponential time”, $\text{DTIME}(2^{o(n)})$. This conjecture accords with the intuition that not only does $P \neq NP$ but actually NP is really at exponential level.

One example of a lower bound was the original paper of Cai and Juedes [CJ01, CJ03] who proved the following definitive result.

Theorem 3 (Cai and Juedes [CJ01, CJ03]). *k -PLANAR VERTEX COVER, k -PLANAR INDEPENDENT SET, k -PLANAR DOMINATING SET, and k -PLANAR RED/BLUE DOMINATING SET cannot be in $O^*(2^{o(\sqrt{k})})$ -FPT unless $\text{FPT} = M[1]$ (or, equivalently, unless ETH fails).*

We remark that Theorem 3 is optimal as all the problems above have been classified as $O^*(2^{O(\sqrt{k})})$ (see e.g. Downey and Thilikos [DTH11])

The obvious connection between subexponential complexity and parameterized complexity classes as formalized by Chen and Grohe [CG07] by constructing an isomorphism, the so-called *miniaturization*, between exponential time complexity (endowed with a suitable notion of reductions) and XP (endowed with FPT reductions) such that the respective notions of tractability correspond, that is, subexponential time on the one and FPT on the other side. Other connections between classical complexity and “canonical” miniaturizations can be found in Downey, Flum, Grohe and Weyer [DFGW07].

2.2 XP-Optimality

There is a new programme akin to the above establishing tight lower bounds on parameterized problems, assuming various non-collapses of the parameterized hierarchies. A powerful example of this is what is called XP optimality. This new programme regards the classes like $W[1]$ as artifacts of the basic problem of proving hardness under reasonable assumptions, and strikes at membership of XP. We illustrate this via INDEPENDENT SET and DOMINATING SET which certainly are in XP. But what’s the best exponent we can hope for for slice k ?

Theorem 4 (Chen et. al [CCFHJKX05]). *The following hold:*

- (i) INDEPENDENT SET cannot be solved in time $n^{o(k)}$ unless $\text{FPT} = M[1]$.
- (ii) DOMINATING SET cannot be solved in time $n^{o(k)}$ unless $\text{FPT} = M[2]$.

More on parameterized intractability and its development over time can be found in the article by Jianer Chen and Iyad Kanj [CK12] in this volume.

3 Positive Techniques

3.1 Bounded Search Trees

A fundamental source of high running times is *branching* in algorithms. A very crude idea to limit the running time is to keep this branching small and a function

of the parameter. For instance, for VERTEX COVER, we can do this as follows. Take any edge $e = vw$, and begin a search tree, by having leaves labeled by v and w , and for each leaf recursively do this for the graphs gotten by deleting any edge covered by v and w respectively. The depth of this process for a k -vertex cover is k and then we can decide if G has a vertex cover in time $O(2^k |G|)$ using this method.

At a certain point in any of these algorithms, you need to appeal to some kind of combinatorics to improve performance. A simple illustration of this idea is that if we can make the search tree smaller than the complete binary tree of length k , then the performance will improve. Notice that, if G has no vertex of degree three or more, then G consists of a collection of cycles, and this is pretty trivial to check. Thus we can assume we have vertices of higher degree than 2. For vertex cover of G we must have either v or *all of its neighbours*, so we create children of the root node corresponding to these two possibilities. The first child is labeled with $\{v\}$ and $G - v$, the second with $\{w_1, w_2, \dots, w_p\}$, the neighbours of v , and $G - \{w_1, w_2, \dots, w_p\}$. In the case of the first child, we are still looking for a size $k - 1$ vertex cover, but in the case of the second child we need only look for a vertex cover of size $k - p$, where p is at least 3. Thus, the bound on the size of the search tree is now somewhat smaller than 2^k . It can be shown that this algorithm runs in time $O(5^{k \setminus 4} \cdot n)$, and in typical graphs, there are lots of vertices of higher degree than 3, and hence this works even faster.

The best algorithms along these lines use more complex branching rules. For example, Niedermeier [Nie02] uses the following branching rules.

BRANCHING RULE VC1:

If there is a degree one vertex v in G , with single neighbour u , then there is a minimum size cover that contains u . Thus, we create a single child node labeled with $\{u\}$ and $G - u$.

BRANCHING RULE VC2:

If there is a degree two vertex v in G , with neighbours w_1 and w_2 , then either both w_1 and w_2 are in a minimum size cover, or v together with *all other neighbours* of w_1 and w_2 are in a minimum size cover.

BRANCHING RULE VC3:

If there is a degree three vertex v in G , then either v or all of its neighbours are in.

We remark that using these three rules, (using a recurrence relation) it can be shown that if there is a solution of size at most k then the size of the corresponding search tree has size bounded above by $O(1.47^k)$. More involved rules of similar ilk exploring the *local structure* of neighbourhoods in graphs, result in the algorithm of Chen et. al. [CKX10] with running time $O(1.2738^k)$ for the branching.

There are a number of problems for which this technique is the only method, or at least the best method, for parameterized algorithms. The method has been particularly successful in computational biology with problems like the

CLOSEST STRING problem [GNR01] and MAXIMUM AGREEMENT FOREST problem [HM07].

In passing I remark that this method is inherently parallelizable and as we see is often used in conjunction with other techniques. The method for VERTEX COVER can be found discussed in [ALSS06].

The paper in this volume by Marx [Ma12] explores the applicability of this technique against other methods like kernelizability discussed in the next section.

3.2 Kernelization

This is again a pretty simply basic idea. If we can make the problem *smaller* then the search will be quicker. This is a *data reduction* or *pre-processing* idea, and is the heart of many heuristics.

Whilst there are variations of the idea below, the simplest version of kernelization is the following.

Definition 3 (Kernelization)

Let $L \subseteq \Sigma^* \times \Sigma^*$ be a parameterized language. A reduction to a problem kernel, or kernelization, comprises replacing an instance (I, k) by a reduced instance (I', k') , called a problem kernel, such that

- (i) $k' \leq k$,
- (ii) $|I'| \leq g(k)$, for some function g depending only on k , and
- (iii) $(I, k) \in L$ if and only if $(I', k') \in L$.

The reduction from (I, k) to (I', k') must be computable in time polynomial in $|I| + |k|$.

There are other notions, where the kernel may be another problem (often “annotated”) or the parameter might increase, but, crucially, the size of the kernel depends *only* on k .

Here are some natural reduction rules for a kernel for VERTEX COVER.

REDUCTION RULE VC1:

Remove all isolated vertices.

REDUCTION RULE VC2:

For any degree one vertex v , add its single neighbour u to the solution set and remove u and all of its incident edges from the graph.

These rules are obvious. Sam Buss (see [DF98]) originally observed that, for a simple graph G , any vertex of degree greater than k must belong to every k -element vertex cover of G (otherwise all the neighbours of the vertex must be included, and there are more than k of these).

This leads to our last reduction rule.

REDUCTION RULE VC3:

If there is a vertex v of degree at least $k+1$, add v to the solution set and remove v and all of its neighbours.

After exhaustively applying these rules, we get to a graph (G', k') , where no vertex in the reduced graph has degree greater than $k' \leq k$, or less than two. Then simple combinatorics shows that if such a reduced graph has a size k vertex cover, its must have size $\leq k^2$. This is the size k^2 kernelization.

Now we can apply the bounded depth search tree rule to this reduced graph, and get an algorithm for vertex cover running in time $O(1.2738^k)k^2$. As observed by Langston and his team in problems in sequence analysis, and articulated by Niedermeier and Rossmanith [NR00] better running times can be obtained by *interleaving* depth-bounded search trees and kernelization. That is, first kernelize, begin a bounded search tree, and the *rekernelize* the children, and repeat. This really does make a difference. In [Nie02] the 3-HITTING SET problem is given as an example. An instance (I, k) of this problem can be reduced to a kernel of size k^3 in time $O(|I|)$, and the problem can be solved by employing a search tree of size 2.27^k . Compare a running time of $O(2.27^k \cdot k^3 + |I|)$ (without interleaving) with a running time of $O(2.27^k + |I|)$ (with interleaving).

In actual implementations there are other considerations such as load sharing amongst processors and the like. We refer to the articles in the Computer Journal special issue concerning practical FPT.

We also remark that there are many strategies of reduction rules to *shrink* the kernel. these include things like *crown reductions* (Abu-Khizam et. al. [ACFLSS04]), and other crown structures (such as N. Abu-Khizam et. al. [AFLS07]) which generalize the notion of a degree 1 vertex having its neighbours in the vertex cover, to more complicated structures which resemble “crowns” attached to the graph⁶. In fact generalizing this to even more complex structures called *protrusions* which have a well-behaved structure of small “treewidth” (we will soon meet in Section 5.1) is an excellent source of theoretically efficient algorithms as evidenced by the “metakernelization” paper Bodlaender et. al. [BFLPST09], though the practicality of such is not at all explored.

Clearly, another game is to seek the smallest kernel. For instance, we know by Nemhauser and Trotter [NT75] a size $2k$ kernel is possible for VERTEX COVER. A natural question is “can we do better?”. As we later see, modulo some complexity considerations, sometimes we can show lower bounds on kernels. (Clearly, if $P = NP$ then all have constant size kernels, so some assumption is needed.) We refer to the site

⁶ Specifically, a *crown* in a graph $G = (V, E)$ consists of an independent set $I \subseteq V$ (no two vertices in I are connected by an edge) and a set H containing all vertices in V adjacent to I . A crown in G is formed by $I \cup H$ iff there exists a size $|H|$ maximum matching in the bipartite graph induced by the edges between I and H , that is, every vertex of H is matched. It is clear that degree-1 vertices in V , coupled with their sole neighbours, can be viewed as the most simple crowns in G . If we find a crown $I \cup H$ in G , then we need at least $|H|$ vertices to cover all edges in the crown. Since all edges in the crown can be covered by admitting at most $|H|$ vertices into the vertex cover, there is a minimum size vertex cover that contains all vertices in H and no vertices in I . These observations lead to the reduction rules based on deleting crowns.

<http://fpt.wikidot.com/fpt-races>

for lots of kernel races.

The state of the art in the theory of kernelization can be found in the article by Daniel Lokshtanov, Neeldhara Misra and Saket Saurabh [LMS12] in this volume.

It is not hard to show that a problem is FPT iff it is kernelizable. However, it is not true that FPT=*polynomial size* kernelizable, and this is discussed in the article by Marx [Ma12] in this volume, along with a future agenda for parameterized complexity.

Another practical technique for establishing parameterized tractability is the following.

3.3 Iterative Compression

This technique was first introduced in a paper by Reed, Smith and Vetta in 2004 [RSV04] and more or less re-discovered by Karp [Ka11]. Although currently only a small number of results are known, it seems to be applicable to a range of parameterized minimization problems, where the parameter is the size of the solution set. Most of the currently known iterative compression algorithms solve *feedback set problems* in graphs, problems where the task is to destroy certain cycles in the graph by deleting at most k vertices or edges. In particular, the K -GRAPH BIPARTISATION problem, where the task is to find a set of at most k vertices whose deletion destroys all odd-length cycles, has been shown to be FPT by means of iterative compression [RSV04]. This had been a long-standing open problem in parameterized complexity theory.

Definition 4 (Compression Routine)

A compression routine is an algorithm that, given a problem instance I and a solution of size k , either calculates a smaller solution or proves that the given solution is of minimum size.

Here is a compression routine for VERTEX COVER. Begin with $(G = (V, E), k)$, we build the graph G vertex by vertex. We start with an initial set of vertices $V' = \emptyset$ and an initial solution $C = \emptyset$. At each step, we add a new vertex v to both V' and C , $V' \leftarrow V' \cup \{v\}$, $C \leftarrow C \cup \{v\}$. We then call the compression routine on the pair $(G[V'], C)$, where $G[V']$ is the subgraph induced by V' in G , to obtain a new solution C' . If $|C'| > k$ then we output NO, otherwise we set $C \leftarrow C'$.

If we successfully complete the n th step where $V' = V$, we output C with $|C| \leq k$. Note that C will be an optimal solution for G .

The compression routine takes a graph G and a vertex cover C for G and returns a smaller vertex cover for G if there is one, otherwise, it returns C unchanged. Each time the compression routine is used it is provided with an intermediate solution of size at most $k + 1$.

The implementation of the compression routine proceeds as follows. We consider a smaller vertex cover C' as a *modification* of the larger vertex cover C . This modification retains some vertices $Y \subseteq C$ while the other vertices $S = C \setminus Y$

are replaced with $|S| - 1$ new vertices from $V \setminus C$. The idea is to try by brute force all $2^{|C|}$ partitions of C into such sets Y and S . For each such partition, the vertices from Y along with all of their adjacent edges are deleted. In the resulting instance $G' = G[V \setminus Y]$, it remains to find an optimal vertex cover that is disjoint from S . Since we have decided to take no vertex from S into the vertex cover, we have to take that endpoint of each edge that is not in S . At least one endpoint of each edge in G' is in S , since S is a vertex cover for G' . If both endpoints of some edge in G' are in S , then this choice of S cannot lead to a vertex cover C' with $S \cap C' = \emptyset$. We can quickly find an optimal vertex cover for G' that is disjoint from S by taking every vertex that is not in S and has degree greater than zero. Together with Y , this gives a new vertex cover C' for G . For each choice of Y and S , this can be done in time $O(m)$, leading to $O(2^{|C|}m) = O(2^k m)$ time overall for one call of the compression routine. With at most n iterations of the compression algorithm, we get an algorithm for κ -VERTEX COVER running in time $O(2^k mn)$.

The parametric tractability of the method stems from the fact that each intermediate solution considered has size bounded by some $k' = f(k)$, where k is the parameter value for the original problem. It works very well with *monotone* problems, where if we get an intermediate no then the answer is definitely no. Note that many minimization problems are not monotone in this sense. For example, a NO instance $(G = (V, E), k)$ for κ -DOMINATING SET can be changed to a YES instance by means of the addition of a single vertex that is adjacent to all vertices in V .

Niedermeier [Nie06] has an excellent discussion of this technique, which would seem to have a lot of applications.

4 Not-Quite-Practical FPT Algorithms

There are a number of distinctive techniques used in parameterized complexity which are “not-quite-practical” FPT algorithms, in the sense that the running times are not feasible in general, but can be in certain circumstances. Additionally, some can be randomized and ones using logical metatheorems can later admit considerable refinement in practice for a *specific* problem. These techniques include color-coding and dynamic programming on bounded width graph decompositions. Since this survey is meant to be brief, I will only allude to these techniques.

4.1 Colour-Coding

This technique is useful for problems that involve finding small subgraphs in a graph, such as paths and cycles. Introduced by Alon et al. [AYZ94], it can be used to derive seemingly efficient randomized FPT algorithms for several subgraph isomorphism problems.

It remains in the “not quite practical” basket due to the large numbers needed to implement it. Here is a brief description of how the method works. We will

apply the problem to k -PATH which seeks to find a (simple) path of k vertices in G . What we do is to *randomly* color the whole graph with k colors, and look for a *colorful* solution, namely one with k vertices of one of each color.

The two keys to this idea are

- (i) we can check for colorful paths quickly.
- (ii) if there is a simple path then the probability that it will have k colors for a random coloring is $\frac{k!}{k^k}$ which is bounded by e^{-k} .

Then, given (i) and (ii), we only need repeat process enough to fast probabilistic algorithm. We prove (i) by using dynamic programming: simply add a vertex v_0 with color 0, connect to those of color 1, then generate the colorful paths of length i starting from v_0 inductively, rather like Dijkstra's algorithm, the running time being $O(k2^k|E|)$.

Theorem 5 (Alon, Yuster and Zwick [AYZ94]). *k -PATH can be solved in expected time $2^{O(k)}|E|$.*

Alon, Yuster and Zwick demonstrated that this technique could be applied to a number of problems of the form asking "is G' a subgraph of G ?" The desired FPT algorithm can now be obtained by a process of derandomization. A *k -perfect family of hash functions* is a family \mathcal{F} of functions (colorings) taking $[n] = \{1, \dots, n\}$ onto $[k]$, such that for all $S \subseteq [n]$ of size k there is a $f \in \mathcal{F}$ whose restriction to S is bijective (colourful). It is known that k -perfect families of $2^{O(k)} \log n$ linear time hash functions. This gives a deterministic $2^{O(k)}|E| \log |V|$ algorithm for k -PATH. More such applications can be found in Downey and Fellows [DF98], and Niedermeier [Nie02, Nie06]. The $O(k)$ in the exponent hides evil, and the derandomization method at present seems far from practical.

Note that the method does not work when applied to things like k -CLIQUE to be shown randomized FPT because (i) above *fails*. The important part of the dynamic programming method was that a path was represented by its beginning v_0 and some vertex v_i , and to extend the path only needed *local knowledge*; namely the colors used so far and v_i . This fails for CLIQUE, and would need $\binom{n}{i}$ at step i in the clique case.

We remark that recent work ([BDFH08, BDFH09]) has shown, assuming a reasonable complexity assumption (namely that the polynomial time hierarchy does not collapse to two or fewer levels), there is no polynomial size kernel for k -PATH. We meet this result in Section 6.

4.2 Bounded Integer Programming

One technique, not discussed in [DF98, FG06], is the use of INTEGER PROGRAMMING in the design of FPT algorithms. This is discussed in Niedermeier [Nie02, Nie06].

Theorem 6 (Lenstra [Le83]). *The integer programming feasibility problem can be solved with $O(p^{\frac{9p}{2}}L)$ arithmetical operations in \mathbb{Z} of $O(p^{2p}L)$ bits in size, where p is the number of variables, and L the number of bits of the input.*

Niedermeier [Nie02, Nie06] gave one example of the use of this method for establishing parametric tractability. He showed that the following problem is FPT.

CLOSEST STRING (parameterized by the number of strings and length)

Input: k strings s_1, \dots, s_k over an alphabet Σ each having length L , and a non-negative integer d .

Parameter: k, L, d .

Question: is there a string s of distance $\leq d$ from s_i for all i ?

We remark that the method's practicality is far from explored. We also refer the reader to Gramm, Niedermeier and Rossmanith [GNR01].

4.3 Bounded Width Metrics

Anyone who has done any course in algorithms has seen various algorithms for planar this and bounded degree, dimension, pathwidth, bandwidth, etc that. Clearly, what is going on is some kind of quest to try to map the boundary of intractability, and using some kind of regularity in the data to get tractability.

Planarity is natural since a road map of a city is more or less planar subject to a few exceptions. One could view the number of exceptions as a parameter, or simply view the every increasing genus as the relevant parameter. Similarly degree. How does the running time vary for the problem at hand as the degree varies.

Two sweeping generalizations of the notions of bounded global parameters are found in the notions of *width metrics*, and in particular through treewidth and local treewidth (defined in the next section). Treewidth is part of the change from *ad hoc* graph theory to structural, topological graph theory which has revolutionized the area in the last decade or so. The following definition is now quite mainstream in modern graph theory.

Definition 5 (Robertson and Seymour [RS86a])

- (a) A tree-decomposition of a graph $G = (V, E)$ is a tree \mathcal{T} together with a collection of subsets T_x (called bags) of V labeled by the vertices x of \mathcal{T} such that $\cup_{x \in \mathcal{T}} T_x = V$ and (i) and (ii) below hold:
- (i) For every edge uv of G there is some x such that $\{u, v\} \subseteq T_x$.
 - (ii) (Interpolation Property) If y is a vertex on the unique path in \mathcal{T} from x to z then $T_x \cap T_z \subseteq T_y$.
- (b) The width of a tree decomposition is the maximum value of $|T_x| - 1$ taken over all the vertices x of the tree \mathcal{T} of the decomposition.
- (c) The treewidth of a graph G is the minimum treewidth of all tree decompositions of G .

The point of the notion is that it is a measure of how treelike the graph is. One can similarly define *path decomposition* where \mathcal{T} must be a path. A tree decomposition is a road map as to how to build the graph. Knowing a tree or

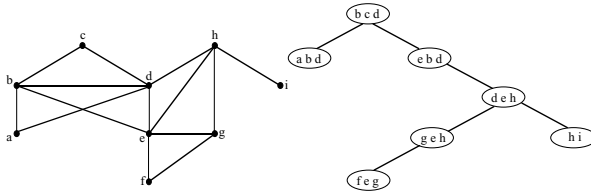


Fig. 1. Example of Tree Decomposition of Width 2

path decomposition of a graph allows for dynamic programming since what is important is the “information flow” across a relatively narrow cut in the graph. Figure 1 gives an example of a tree decomposition of width 2.

Authors often discovered that intractable problems became tractable if the problems were restricted to say, “outerplanar” graphs. As we have seen, such restriction is not purely an academic exercise since, in many practical situations, the graphs that arise do not in fact demonstrate the full pathology of the class of all graphs. Families of graph that have been studied which turn out to have bounded treewidth include Almost Trees (k) (width $k + 1$), Bandwidth k (width k), Cutwidth k (width k), Planar of Radius k (width $3k$), Series Parallel (width 2), Outerplanar (width 2), Halin (width 3) k -Outerplanar (width $3k - 1$), Chordal with Maximum Clique Size k (width $k - 1$), and many others.

4.4 Algorithms for Graphs of Bounded Treewidth

We sketch how to run algorithms on graphs of bounded treewidth. This can be viewed as *dynamic programming* a very important algorithmic technique. Many classical problems are known to be algorithmically infeasible on general graphs (in that they take exponential time) but become polynomial time when restricted to some bounded treewidth class. A good introduction to this technique is Binstock and Langston [BL95], Bodlaender and Kloks [BK96], Bodlaender and Koster [BoK08], or Babette de Fluiter [deF70]. I am always surprised when I meet people who are unaware of this technique, since it has been around so long. But for completeness it seems worthwhile to describe the method.

We describe the technique for the INDEPENDENT SET. Whilst this problem is classically $W[1]$ -complete, in the case of graphs of bounded treewidth, we can give a linear time algorithm. So suppose that we have a tree decomposition of G . Consider the one below

Now what we will do is to use *tables* to grow the independent set up the tree starting at the leaves of the decomposition. Notice that once a vertex leaves the bags it will never come back, and hence we don’t really need to keep track of its effect. Thus we can work with tables corresponding to all the subsets of the current bag and need only consider independent sets I relative to the current bag. That is, we would consider all the subsets of the bag and see how the size of independent sets relative to in the information flow across the boundary.

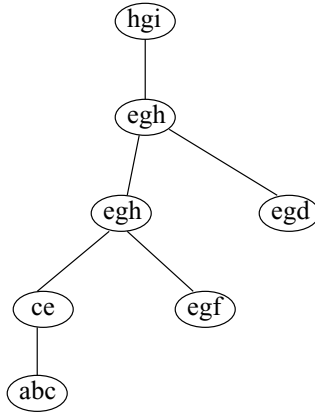
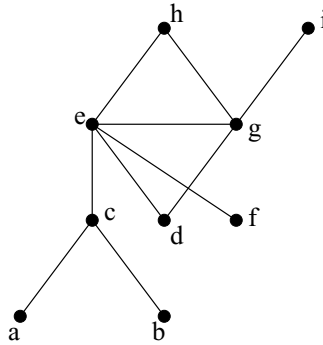


Fig. 2. A tree decomposition

Thus at the leaf corresponding to the triangle abc we could have the table below. Here, for instance ab denotes the set $\{a, b\}$, meaning that the independent set should contain both a and b , and would have cardinality 2, and bc corresponds to $\{b, c\}$ and this entry has a line since $\{b, c\}$ is *not* an independent set.’

\emptyset	a	b	c	ab	ac	bc	abc
0	1	1	1	2	-	-	-

The table for the next box would have only 4 columns since there are only 4 subsets of $\{c, d\}$ and we consider maximal independent sets I containing the specified subset

\emptyset	c	e	ce
2	1	3	-

The first entry has value 2 since this means neither of c or d in included in this I and hence we take the maximal one below, namely 2.

The second entry says that I must contain c , and this corresponds to the entry from below containing c , which is 1. The third one corresponds to the

independent set containing e and *not* c , and hence we get 3, by choosing the largest one with this property from below (namely the entry for ab contains no c , and this together with e gives $1 + 2 = 3$.)

Finally, the last entry is 0 since I would need to contain both c and d and no independent set has this property.

The rest of the table is filled in similarly. With pointers you can also keep track of the relevant independent set.

The next table is a join node

\emptyset	e	g	h	eg	eh	gh	egh
0	1	1	1	2	-	-	-

This table corresponds to $I \cap \{e, g, h\}$ being the specified set. The first entry corresponds to I having nothing from this set. It has value 3 since we get that from the 2 on the left and 1 on the right. (We must add here.)

The second entry corresponds to the intersection being $\{e\}$, meaning that it must have e and cannot have h or g . Note e is present in both branches and hence we get 3. Next is g with e and h not present. There are 2 from the left (corresponding to the \emptyset), and g is present on the right so we take its entry giving 3. the next is h and neither e or g , giving 2 from the left branch, 2 from the right (using f since h is not present) and h itself; giving the total of 4. etc

The final table looks like:

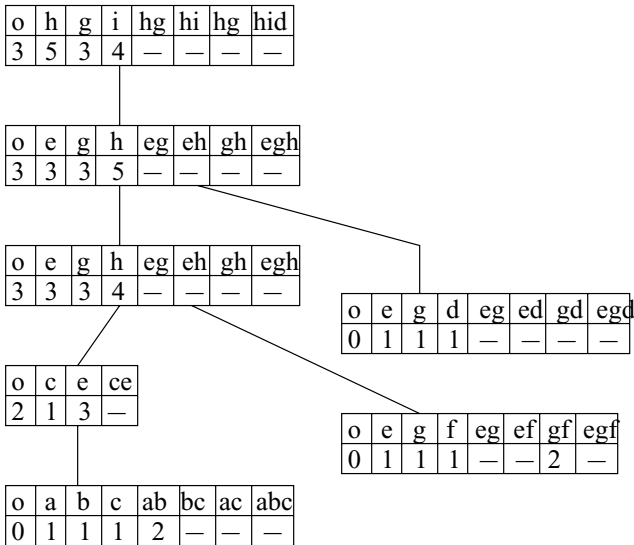


Fig. 3. Dynamic programming

We remark that a sly feature we have not mentioned here is that to run this method, *we need a tree decomposition for the graph G* . It turns out that for a fixed t there is a *linear time algorithm* determining if G has width t , and then finds the tree decomposition of G should the graph actually have one. However, the algorithm has really terrible constants and there is no actually feasible one for this problem. For more on this we refer the reader to Bodlaender [Bod93, Bod96], Bodlaender-Kloks [BK96] and Bodlaender's web site. John Fouhy in his MSc Thesis [Fou03] looked at computational experiments for treewidth heuristics. (See www.mcs.vuw.ac.nz/~downey/students.html.) There is a lot of work to be done here.

Actually, this whole process can be implemented by automata acting on trees. We refer the reader to Downey and Fellows [DF98], Fellows and Langston [FL89], or Abrahamson-Fellows [AF93].

Treewidth is the archetype of a number of graph width metrics⁷ naturally arise in this context which restrict the inherent complexity of a graph in various senses. The idea here is that a useful width metric should admit efficient algorithms for many (generally) intractable problems on the class of graphs for which the width is small. This leads to consideration of these measures from a parameterized point of view. The corresponding naturally parameterized problem has the following form:

Let $w(G)$ denote any measure of graph width.

Instance: A graph $G = (V, E)$.

Parameter: A positive integer k .

Question: Is $w(G) \leq k$?

The hope is that this problem is FPT, and then, equipped with the relevant decomposition, we can run algorithms. Another important example of a width metric is *cliquewidth* where a clique decomposition works as follows. We inductively define graphs using sets of $k + 1$ colours and *parse* operators as follows. The first operator is c_i which says "create a vertex of colour i ." The second one is $j(i, j)$ which says "join all vertices of colour i to all vertices of colour j ." The is d_j which says "form the disjoint union of all graphs constructed so far", and finally $r(i, j)$ which says "recolour all vertices of colour i to j ." The width of the decomposition is the smallest number of colours necessary minus 1. For example, a clique has with 1 : to make a clique of size n , create a vertex coloured 1, and then one of colour 2, apply $j(1, 2)$, then $r(1, 2)$, then create a new vertex of colour 1, and repeat enough times. It is not hard to show that if G has bounded treewidth then it also has bounded cliquewidth.

It is unknown if CLIQUEWIDTH is FPT or $W[1]$ hard. It is known to be NP-complete for k varying by Fellows, Rosamond, Rotics and Szeider [FRRS06, FRRS09]. There is a lot of evidence that it is parametrically hard such as Fomin et. al. [FGLS10].

⁷ And have been used in other settings such as matroids where the width corresponds to the dimension of the intersection of "subspaces" in some decomposition, See, for example Hlineny Whittle [HW06].

However, sometimes we are lucky and are supplied with a clique decomposition. The point is that *given* a clique decomposition we can run linear time algorithms for some problems. Thus the *certificated* problem where we input G and its clique decomposition, and ask questions is often FPT.

For much more on treewidth and other width parameters, we refer the article in this volume by Bodlaender [Bod12].

4.5 Logic

One source of (often impractical) FPT results is the use of metatheorems from logic. The idea is that to feed in some logical description of the problem, and use some algorithmic machine to given an FPT algorithm for it. The generality of the methodology often means that the algorithms obtained have large constants, and implementations need fine tuning. Sometimes you can use methods from (parameterized) complexity to show that the large constants *can't* be removed, but more on this later.

We tend to look at problems defined in *first-order logic* and *monadic second-order logic*. We remind the reader that first order logic uses individual variables and form the logic by the following rules.

1. *Atomic formulas:* $x = y$ and $R(x_1, \dots, x_k)$, where R is a k -ary relation symbol and x, y, x_1, \dots, x_k are individual variables, are FO-formulas.
2. *Conjunction, Disjunction:* If ϕ and ψ are FO-formulas, then $\phi \wedge \psi$ is an FO-formula and $\phi \vee \psi$ is an FO-formula.
3. *Negation:* If ϕ is an FO-formula, then $\neg\phi$ is an FO-formula.
4. *Quantification:* If ϕ is an FO-formula and x is an individual variable, then $\exists x \phi$ is an FO-formula and $\forall x \phi$ is an FO-formula.

First order logic allows for the description of local behaviour of structures. For instance to say that a graph has an independent set of size k ,

$$\exists x_1 \dots x_k \bigwedge_{1 \leq i < j \leq k} \neg E(x_i, x_j),$$

where $E(x, y)$ denoted the edge relation on the graph.

For monadic second order logic we add *set* variables, one for each subset of vertices in the graph. Formulas of monadic second-order logic (MSO) are formed by the rules for FO and the following additional rules:

1. *Additional atomic formulas:* For all set variables X and individual variables y , Xy is an MSO-formula.
2. *Set quantification:* If ϕ is an MSO-formula and X is a set variable, then $\exists X \phi$ is an MSO-formula, and $\forall X \phi$ is an MSO-formula.

We can state that a graph is k -colorable using an MSO-formula,

$$\exists X_1 \dots \exists X_k \left(\forall x \bigvee_{i=1}^k X_i x \wedge \forall x \forall y \left(E(x, y) \rightarrow \bigwedge_{i=1}^k \neg (X_i x \wedge X_i y) \right) \right)$$

Actually I am being sloppy here as there are variations of the meaning of this depending on whether individual and set variables are allowed for edges; and if so this is denoted by MS_2 . There is a whole industry here devoted to the use of logic in algorithmic for computer science, and I will concentrate on MS_2 only to give the flavour of the methodology. For more on this see [DF98, FG06].

Now, whilst model checking for even first order formulae is already PSPACE complete, we have the following.

Theorem 7 (Courcelle 1990). *The model-checking problem for MS_2 restricted to graphs of bounded treewidth is linear-time fixed-parameter tractable.*

Thus, the INDEPENDENT SET problem above can easily be obtained from Courcelle's Theorem by simply writing a formula in monadic second order logic describing that G has a k -independent set.

Actually Courcelle's Theorem is true for a mild extension of MS_2 called MS_2^+ where certain "counting" relations are added. The result gives the flavour of the methods from logic. If we have a suitably restricted class of graphs, then model checking becomes FPT. Examples include first order logic for families of graphs of bounded *local treewidth*. This is the notion of how fast the treewidth grows in a neighbourhood of a vertex of any graph in the family. This is called bounded if there is a function f , such that for all n , the treewidth of the n -neighbourhood of a vertex of any member of the family is bounded by $f(n)$. Examples of classes of graphs that have bounded local treewidth include graphs of bounded treewidth (naturally), graphs of *bounded degree*, *planar* graphs, and graphs of *bounded genus*.

Theorem 8 (Frick and Grohe [FrG01]). *Parameterized problems that can be described as model-checking problems for FO are fixed-parameter tractable on classes of graphs of bounded local treewidth.*

There are other notions of bounded with where this works. In general, things like treewidth, cliquewidth, and any other of these metrics indicate the the member of of the graph family will be built by certain inductive methods. Such methods clearly have reflections in reality when we consider how, for example, computer chips are designed. It is not surprising that such inductive families tend to have better algorithmics than general graphs. For more on this we refer the reader to [GK11].

5 Exotica, WQO Theory

In this last section we look at exotic methods for proving problems to be FPT. These tend to only give membership of FPT and no practical algorithms. As discussed in Downey [Do12] and Langston [La12] in this Festschrift, this material is the parent of parameterized complexity, in a sense made clear in those papers, and emanating from material such as [FL87, FL88].

A *quasi-ordering* on a set S is a reflexive transitive relation on S . We will usually represent a quasi-ordering as \leq_S or simply \leq when the underlying set S is clear. Let $\langle S, \leq \rangle$ be a quasi-ordered set. We will write $x < y$ if $x \leq y$ and $y \not\leq x$, $x \equiv y$ if $x \leq y$ and $y \leq x$ and finally $x|y$ if $x \not\leq y$ and $y \not\leq x$. Note that if $\langle S, \leq \rangle$ is a quasi-ordered set then S/\equiv is partially ordered by the quasi-order induced by \leq . Recall that a *partial ordering* is a quasi-ordering that is also antisymmetric.

Definition 6 (Ideal and Filter). Let $\langle S, \leq \rangle$ be a quasi-ordered set. Let S' be a subset of S .

- (i) We say that S' is a *filter* if it is closed under \leq upwards. That is, if $x \in S'$ and $y \geq x$ then $y \in S'$. The filter generated by S' is the set $F(S') = \{y \in S : \exists x \in S'(x \leq y)\}$.⁸
- (ii) We say that S' is a (lower) *ideal* if S' is \leq closed downwards. That is if $x \in S'$ and $y \leq x$ then $y \in S'$. The ideal generated by S' is the set $I(S') = \{y \in S : \exists x \in S'(x \geq y)\}$.
- (iii) Finally, if S' is a filter (an ideal) that can be generated by a finite subset of S' then we say that S' is *finitely generated*.

We will need some distinguished types of sequences of elements.

Definition 7. Let $\langle S, \leq \rangle$ be a quasi-ordered set. Let $A = \{a_0, a_1, \dots\}$ be a sequence of elements of S . Then we say the following.

- (i) A is *good* if there is some $i < j$ with $a_i \leq a_j$.
- (ii) A is *bad* if it is not good.
- (iii) A is an *ascending chain* if for all $i < j$, $a_i \leq a_j$.
- (iv) A is an *antichain* if for all $i \neq j$ $a_i|a_j$.
- (v) $\langle S, \leq \rangle$ is *Noetherian* if S contains no infinite (strictly) descending sequences. (i.e. there is no sequence $b_0 > b_1 > b_2 \dots$)
- (vi) $\langle S, \leq \rangle$ has the *finite basis property* if for all subsets $S' \subseteq S$, $F(S')$ is *finitely generated*.

The following result is relevant to our work.

Theorem 9 (Folklore, after Higman [Hi52]). Let $\langle S, \leq \rangle$ be a quasi-ordered set. The following are equivalent.

- (i) $\langle S, \leq \rangle$ has no bad sequences.
- (ii) Every infinite sequence in S contains an infinite chain.
- (iii) $\langle S, \leq \rangle$ is Noetherian and S contains no infinite antichain.
- (iv) $\langle S, \leq \rangle$ has the finite basis property.

Definition 8 (Well Quasi-Ordering). Let $\langle S, \leq \rangle$ be a quasi-ordering. If $\langle S, \leq \rangle$ satisfies any of the characterizations of Theorem 9, then we say that $\langle S, \leq \rangle$ is a *well quasi-ordering (WQO)*.

⁸ Sometimes, filters are called *upper ideals*.

The reader might well wonder what any of this abstract pure mathematics has to do with algorithmic considerations. The key is provided by the finite basis characterizations of a WQO. Suppose that \leq is the relevant quasi-ordering and for a fixed x the question “ Q_y : Is $x \leq y$?” is FPT (resp. polynomial time). Then for a WQO, if F is a filter, then F has a finite basis $\{b_1, \dots, b_n\}$. Then to decide if $y \in F$ we need only ask “ $\exists i \leq n (b_i \leq_S y)$?” That is, membership of each filter is FPT (resp. polynomial time) (even though we don’t know the finite basis!).

Often the argument is phrased in terms of *obstruction sets*. Let $\langle S, \leq \rangle$ be a quasi-ordering. Let I be an ideal of $\langle S, \leq \rangle$. We say that a set $O \subseteq S$ forms an *obstruction set* for I if

$$x \in I \text{ iff } \forall y \in O (y \not\leq x).$$

That is O is an obstruction set for I if I is the complement of $F(O)$. The WQO principle says all ideals have finite obstruction sets.

So here is our new engine for demonstrating that problems are in P . Prove that the problem is characterized by a WQO with a finite obstruction set in a quasi-ordering with Q_y in P .

The best known example of an obstruction set is provided by topological ordering.

Definition 9 (Topological Ordering). *A homeomorphic or topological embedding of a graph $G_1 = (V_1, E_1)$ in a graph $G_2 = (V_2, E_2)$ is an injection from vertices V_1 to V_2 with the property that the edges E_1 are mapped to disjoint paths of G_2 . (These disjoint paths in G_2 represent possible subdivisions of the edges of G_1 .) The set of homeomorphic embeddings between graphs gives a partial order, called the topological order. We write $G_1 \leq_{top} G_2$.*

While topological ordering is not a WQO, there are a number of important finite basis results. The most famous is the following (which was independently discovered by Pontryagin).

Theorem 10 (Kuratowski’s Theorem, Kuratowski [Ku30]). *The graphs $K_{3,3}$ and K_5 of Figure 4 form an obstruction set for the ideal of planar graphs in the topological ordering.*

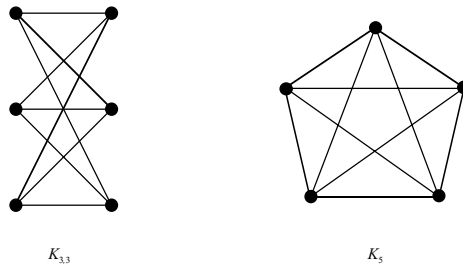


Fig. 4. Obstructions for planarity

Very recently, Grohe, Marx, Kawarabayashi, and Wollan,[GMKW11] proved that \leq_{top} is FPT, with $G \leq_{top} H \iff O(|H|^3)$ for a fixed G . A consequence of that result is that any filter with a finite basis in the topological quasi-ordering has an FPT membership algorithm. The [GMKW11] argument is very difficult, and the algorithm has horrendous constants. We remark that graphs of pathwidth 2 are well-quasi-ordered by \leq_{top} , and hence those graphs have FPT membership for any ideal.

A generalization of topological ordering is much more amenable to being a WQO. An equivalent formulation of \leq_{top} is the following. $G \leq_{top} H$ iff G can be obtained from H by a sequence of the following two operations.

- (i) (deletions) Deleting vertices or edges.
- (ii) (degree 2 contractions) The *contraction* of an edge xy in a graph W is obtained by identifying x with y . A contraction has degree 2 iff one of x or y has degree 2.

The *minor ordering* is a generalization of \leq_{top} is obtained by relaxing the degree 2 requirement in (ii).

Definition 10 (Minor Ordering). *We say that G is a minor of H if G can be obtained from H by a sequence of deletions and contractions. We write $G \leq_{minor} H$.*

An example of the minor ordering is given in Figure 5

The proof of Kuratowski’s Theorem also gives the following.

Theorem 11 (Kuratowski’s Theorem (II)). *$K_{3,3}$ and K_5 are an obstruction set for the ideal of planar graphs under \leq_{minor} .*

The way to think of \leq_{minor} is to think of $G \leq_{minor} H$ as taking $|G|$ many collections C_i of connected vertices of H , and coalescing each collection C_i to a single vertex, then H being topologically embeddable into the new coalesced graph, so that the edges of G become disjoint paths from coalesced vertices. Sometimes this is called the “folio” definition of the minor ordering. Figure 6 below demonstrates this idea.

Notice that H has no vertices of degree 4 and hence there can be no topological embedding of G into H .

Wagner [Wa37] conjectured the following.

Wagner’s Conjecture:*Finite graphs are well quasi-ordered by the minor ordering.*

Notice that graphs of genus $\leq g$ for a fixed g form an ideal in the minor ordering. Hence a consequence of Wagner’s conjecture is a Kuratowski Theorem for surfaces. *Graphs of genus $\leq g$ have a finite obstruction set.* One of the triumphs of 20th century mathematics is the following great theorem of Neil Robertson and Paul Seymour.

Theorem 12 (Graph Minor Theorem, Robertson and Seymour). *Wagner’s Conjecture holds: Finite graphs are well quasi-ordered by the minor ordering.*

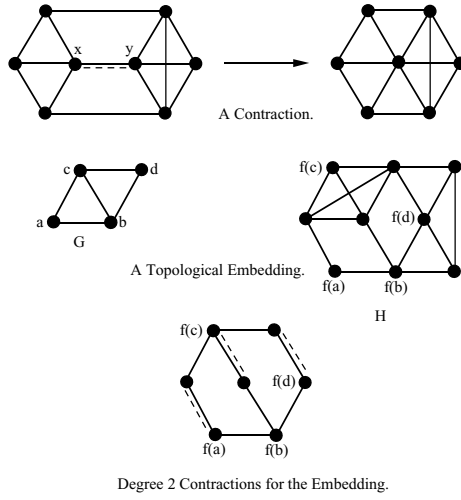


Fig. 5. A contraction and a Topological Embedding

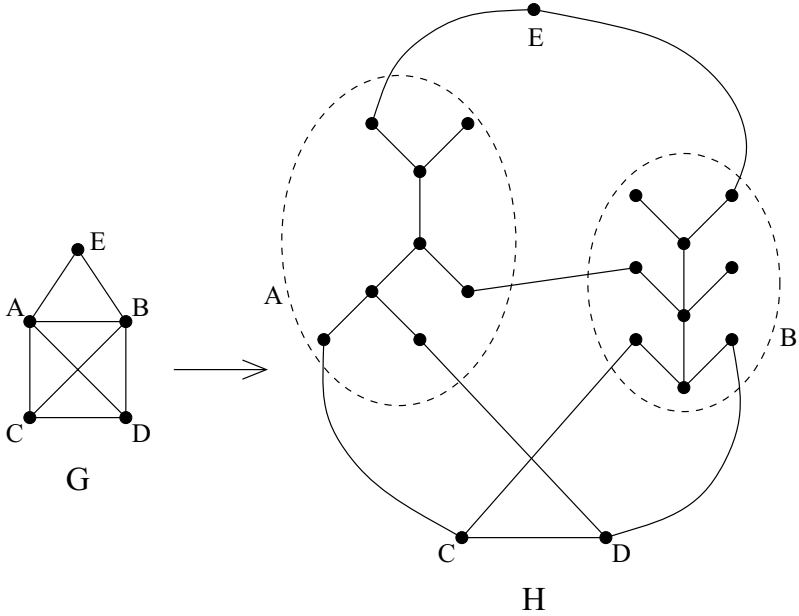


Fig. 6. The folio definition

Remarkably, Robertson and Seymour showed that $x \leq_{\text{minor}} y$ is $O(|y|^3)$ for a fixed x . Thus every minor ideal has a cubic time recognition algorithm!

For example, the problem of determining if a graph has genus k for a fixed k becomes FPT immediately. This is, of course, not the best algorithm, and a constructive linear time FPT one was given by Mohar [Mo99].

Multiple applications of this metatheorem can be obtained by the following. We call a parameter p of graphs *treewidth bounded* if the treewidth of G is bounded by $f(p(G))$ for some computable f . We say that p is MSO definable if $\{G : p(G) \leq k\}$ is monadic second order definable by a formula ϕ_k for each fixed k .

Theorem 13 (Adler, Grohe and Kreutzer [AGK08]). *If p is a treewidth bounded MSO definable parameter, then the obstruction set for $\{G : p(G) \leq k\}$ is finite and bounded by $g(k)$ for some computable g . Thus if p is a treewidth bounded minor closed graph parameter, then checking $p(G) \leq k$ is constructively⁹ in FPT with time bound $g(k) \cdot n$.*

5.1 Protrusions

Using this notion we can prove a structural lemma about the structure of graphs involving what are called *protrusions*. In my mind this is an extension of the concept of a crown.

Definition 11 (Bodlaender et. al. [BFLPST09]). *Given a graph G , $X \subset V(G)$ is called an r -protrusion if the treewidth of $G[X]$ is $\leq r$ and the boundary of X intersection G is $\leq r$.*

Protrusions are good because they allow for efficient kernelizations in the same way that crowns do. Here is an archetypical theorem.

Theorem 14 (Bodlaender et. al. [BFLPST09]). *If P is a problem of finite integer index, there is a computable function f and an algorithm which, given an instance (G, k) and an r -protrusion X of size at least $f(r)$ produces an instance (G^*, k^*) such that $|V(G^*)| < |V(G)|$, $k^* \leq k$ and $(G, k) \in P$ iff $(G^*, k^*) \in P$. Furthermore this runs in time $O(|X|)$.*

The idea here is that we can apply this iteratively to show that methods from the graph minor project can be used efficiently, at least insofar as the side of kernels is concerned.

Algorithms based on the structural idea of finding protrusions and kernelizing has seen a lot of development here. In the next section we will give one illustration, and be content with that for this basic introduction.

⁹ Given the definition of FPT we have used this seems a strange statement. However, simply applying the graph minor machinery establishes that some problem has a cubic time algorithm and we don't know what it is. To know it requires knowledge of the relevant obstruction set. Hence we usually get FPT results for a class known as nonuniform FPT.

5.2 Bidimensionality

This is an area that grew from results about *excluded minor* theorems. Excluded minor theorems are of the kind that say if we exclude a certain graph or graphs from being a minor of a members of that family, then that family is well-behaved. It is one of the underlying intuitions for the Robertson Seymour theory. The archetype result is the following

Theorem 15 (Robertson and Seymour). *For every $n > 0$ there is a c_n such that every graph of treewidth $\geq c_n$ has an n -grid as a minor.*

This result was extended as follows.

Theorem 16 (Demaine and Hajiaghayi [DeH08]). *For every fixed H there is a c_H such that every H -minor-free graph or treewidth $\geq c_H \cdot n$ has an n -grid as a minor.*

This result can be seen as the first part of a theory called *bidimensionality* theory. This is again a large undertaking and we refer the reader to Demaine and Hajiaghayi [DeH08] for details of the theory.

Here is one example. We say a graph parameter p is *minor bidimensional* if p is closed under minors and p evaluated on the k -grid is $\Omega(k^2)$. A problem Π is called *subgraph separable* if its solutions can be described in terms of vertex subsets of the input graph, and there is some constant d such that for each G and $S \subseteq V(G)$, every optimal solution Z for G , every union H of some subsets of connected components of $G \setminus S$, and every optimal solution Z' for H , we have

$$|Z'| - d|S| \leq |Z \cap H| \leq |Z'| + d|S|.$$

The conditions above tend to be relatively easy to apply. The relevant structural lemma is the following.

Lemma 1 (Separation Lemma). *Suppose Π is a problem that is subgraph separable and minor-bidimensional. Fix a graph H . Then there is a constant $c_{H,\Pi}$ such that for every H -minor-free graph $(G, k) \in \Pi$, there is a subset $S \subset V(G)$, with $|S| = O(k)$ and the treewidth of $G[V \setminus S] \leq c_{H,\Pi}$.*

The lemma allows us to kernelize by using protrusion like actions on the non-core part of the graph. Skipping details, this allows us to show the following.

Theorem 17 (Demaine, Fomin, Hajiaghayi and Thilikos). *Every subgraph separable minor-bidimensional problem Π with finite index has a linear kernel on graphs excluding some fixed graph as a minor.*

Further details and applications of these ideas are beyond the scope of this basic survey and we refer the reader to [DeH08, GK11] for further details of bidimensionality theory and other algorithmic metatheorems.

For much more on the methods for constructivising results on graph minors, we refer to the article by Dimitrios Thilikos [T12] in this volume.

6 Limitations and Lower Bounds

Many of the results above gave FPT algorithms but an analysis of the algorithms reveal very bad running times. For many of these if the unparameterized problem is NP complete then they would have a polynomial running time. However, assuming that $NP \neq P$, or something akin to that, we can ask if these running times can be improved. Similarly, we have seen that kernelization to a small kernel is a valuable way to generate practical algorithms (although the kernels and algorithms are far from practical in the case of the algorithms from bidimensionality theory). Again we can ask the same question.

The combinatorics of FPT are sensitive to the issues of polynomial time, and can often be used in this way.

For example, we have seen that Frick and Grohe proved that deciding first-order statements is FPT for every fixed class of graphs of bounded local treewidth. Courcelle shows a similar result for MS_2 for graphs of bounded treewidth. In each case the algorithmic metatheorem gives algorithms where each alternation of quantifier (roughly) gives another power of two in towers of powers of two for the constants. In the case of, for example, local treewidth Frick and Grohe [FrG02] prove such towers of two's cannot be removed unless $W[P]=FPT$. Similar results were established by Flum and Grohe for treewidth based around $P \neq NP$.

In terms bounds for kernelizations, there has been a lots of progress in the last few years. Bodlaender, Downey, Fellows and Hermelin [BDFH08, BDFH09] gave general methods for establishing that classes of problems (distillable) did not have polynomial kernels assuming certain unlikely things don't happen to complexity classes. For example, using a lemma of Fortnow and Santhanam [FS11] they showed that no Or-compositional parameterized problem can have a polynomial sized kernel assuming $co-NP \not\subseteq NP/Poly$. Here we refer to [BDFH09, CFM11, FS11, BTY08, HKSWWta] for more details.

7 Left Out

I have left out many things, likely close to various researcher's hearts. For example, there is work on parameterized counting (McCartin [McC06] and Flum and Grohe [FG02b]) where we count the number of paths of length k to define, for instance, $\#W[1]$. One nice theorem here is the following.

Theorem 18 (Flum and Grohe [FG02b]). *Counting the number of cycles of size k in a bipartite graph is $\#W[1]$ -complete.*

This result can be viewed as a parameterized analog of Valiant's theorem on the permanent. Another area is parameterized randomization, such as Downey, Fellows and Regan [DFR98], and Müller [Mu06, Mu08], but here problems remain. Parameterized approximation looks at questions like: Is it possible to have an FPT algorithm which, on parameter k , either outputs a size $2k$ dominating set for G , or says no dominating set of size k ? Such algorithms famously exist for BIN PACKING and don't exist for most natural $W[P]$ complete problems. Here we

refer to Downey, Fellows, McCartin and Rosamond [DFMR08] and Eickmeyer, Grohe and Grüber [EGG08] for more details. We have left out discussions of parameterizing above guaranteed values such as Mahajan and Raman [MR99], plus discussions of the breadth of applications. For the last, we can only point at the relevant books, and the *Computer Journal* issues [DFL08]. In this volume we do look at areas such as artificial intelligence (Gaspers and Szeider [GS12]), biology (Stege [St12]), cryptography (Koblitz [Ko12]), and social choice (Betzler et al. [BBCN12]). But of course, there are many other applications.

There are many other important topics such as implementations, connections with exact algorithms, connections with classical complexity and the like. Space limitations preclude this material being included. Hopefully this brief survey gives the reader the ability to appreciate the articles of this Festschrift.

References

- [ACFLSS04] Abu-Khazam, F., Collins, R.L., Fellows, M.R., Langston, M.A., Sutera, W.H., Symons, C.T.: Kernelization Algorithms for the Vertex Cover Problem: Theory and Experiments. In: Proceedings of the 6th ALENEX 2004, pp. 62–69 (2004)
- [AFLS07] Abu-Khazam, F., Fellows, M., Langston, M., Sutera, W.: Crown Structures for Vertex Cover Kernelization. *Theory Comput. Syst.* 41(3), 411–430 (2007)
- [ALSS06] Abu-Khazam, F., Langston, M., Shanbhag, P., Symons, C.: Scalable Parallel Algorithms for FPT Problems. *Algorithmica* 45, 269–284 (2006)
- [ADF93] Abrahamson, K., Downey, R., Fellows, M.: Fixed Parameter Intractability II (Extended Abstract). In: Enjalbert, P., Wagner, K.W., Finkel, A. (eds.) STACS 1993. LNCS, vol. 665, pp. 374–385. Springer, Heidelberg (1993)
- [ADF95] Abrahamson, K., Downey, R., Fellows, M.: Fixed Parameter Tractability and Completeness IV: On Completeness for $W[P]$ and PSPACE Analogs. *Annals of Pure and Applied Logic* 73, 235–276 (1995)
- [AF93] Abrahamson, K., Fellows, M.: Finite Automata, Bounded Treewidth and Wellquasiordering. In: Graph Structure Theory. Contemporary Mathematics Series, vol. 147, pp. 539–564. American Mathematical Society (1993)
- [AGK08] Adler, I., Grohe, M., Kreutzer, S.: Computing excluded minors. In: Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2008), pp. 641–650 (2008)
- [AR01] Alekhovich, M., Razborov, A.: Resolution is Not Automatizable Unless $W[P]$ is Tractable. In: Proc. of the 42nd IEEE FOCS, pp. 210–219 (2001)
- [AYZ94] Alon, N., Yuster, R., Zwick, U.: Color-Coding: A New Method for Finding Simple Paths, Cycles and Other Small Subgraphs Within Large Graphs. In: Proc. Symp. Theory of Computing (STOC), pp. 326–335. ACM (1994)
- [Ar96] Arora, S.: Polynomial Time Approximation Schemes for Euclidean TSP and Other Geometric Problems. In: Proceedings of the 37th IEEE Symposium on Foundations of Computer Science, pp. 2–12 (1996)
- [Ar97] Arora, S.: Nearly Linear Time Approximation Schemes for Euclidean TSP and Other Geometric Problems. In: Proc. 38th Annual IEEE Symposium on the Foundations of Computing (FOCS 1997), pp. 554–563. IEEE Press (1997)
- [Baz95] Bazgan, C.: Schémas d’approximation et complexité paramétrée, Rapport de stage de DEA d’Informatique à Orsay (1995)

- [BBCN12] Betzler, N., Bredereck, R., Chen, J., Niedermeier, R.: Studies in Computational Aspects of Voting – a Parameterized Complexity Perspective. In: Bodlaender, H.L., et al. (eds.) *Fellows Festschrift*. LNCS, vol. 7370, pp. 318–363. Springer, Heidelberg (2012)
- [BL95] Bienstock, D., Langston, M.: Algorithmic Implications of the Graph Minor Theorem. In: Ball, M.O., Magnanti, T.L., Monma, C.L., Nemhauser, G.L. (eds.) *Handbook of Operations Research and Management Science: Network Models*, pp. 481–502. North Holland (1995)
- [Bod93] Bodlaender, H.L.: A linear time algorithm for finding tree-decompositions of small treewidth. In: *Proceedings of the 25th ACM Symposium on Theory of Computing*, pp. 226–234 (1993)
- [Bod96] Bodlaender, H.L.: A linear time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal on Computing* 25, 1305–1317 (1996)
- [Bod12] Bodlaender, H.L.: Fixed-Parameter Tractability of Treewidth and Pathwidth. In: Bodlaender, H.L., et al. (eds.) *Fellows Festschrift*. LNCS, vol. 7370, pp. 196–227. Springer, Heidelberg (2012)
- [BDFH08] Bodlaender, H.L., Downey, R., Fellows, M., Hermelin, D.: On Problems without Polynomial Kernels (Extended Abstract). In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfssdóttir, A., Walukiewicz, I. (eds.) *ICALP 2008, Part I*. LNCS, vol. 5125, pp. 563–574. Springer, Heidelberg (2008)
- [BDFH09] Bodlaender, H.L., Downey, R., Fellows, M., Hermelin, D.: On Problems without Polynomial Kernels. *Journal of Computing and System Sciences* 75(8), 423–434 (2009)
- [BDFHW95] Bodlaender, H.L., Downey, R., Fellows, M., Hallett, M., Wareham, H.T.: Parameterized Complexity Analysis in Computational Biology. *Computer Applications in the Biosciences* 11, 49–57 (1995)
- [BFH94] Bodlaender, H.L., Fellows, M.R., Hallett, M.T.: Beyond NP-completeness for Problems of Bounded Width: Hardness for the W Hierarchy. In: *Proc. ACM Symp. on Theory of Computing (STOC)*, pp. 449–458 (1994)
- [BFLPST09] Bodlaender, H.L., Fomin, F., Lokshtanov, D., Pennick, E., Saurabh, S., Thilikos, D.: (Meta)kernelization. In: *50th IEEE FOCS* (2009)
- [BK96] Bodlaender, H.L., Kloks, T.: Efficient and constructive algorithms for the pathwidth and treewidth of graphs. *Algorithms* 21, 358–402 (1996)
- [BoK08] Bodlaender, H.L., Koster, A.: Combinatorial optimization on graphs of bounded treewidth. *The Computer Journal* 51, 256–269 (2008)
- [BTY08] Bodlaender, H.L., Thomasse, S., Yeo, A.: Analysis of data reduction, transformations give evidence for non-existence of polynomial kernels, Tech. Rept, UU-CS-2008-030, Utrecht Univ. (2008)
- [BPR01] Bonnet, M., Pitazzi, T., Raz, R.: On Interpolation and Axiomatization for Frege Systems. *SIAM J. Comput.* 29, 1939–1967 (2000)
- [LeC96] Cai, L.: Fixed-parameter tractability of graph modification problems for hereditary properties. *Information Processing Letters* 58(4), 171–176 (1996)
- [CC97] Cai, L., Chen, J.: On Fixed-Parameter Tractability and Approximability of NP-Hard Optimization Problems. *Computer and Systems Sciences* 54, 465–474 (1997)
- [CCDF96] Cai, L., Chen, J., Downey, R.G., Fellows, M.R.: On the Parameterized Complexity of Short Computation and Factorization. *Arch. for Math. Logic* 36, 321–337 (1997)
- [CFJR07] Cai, L., Fellows, M., Juedes, D., Rosamond, F.: The complexity of polynomial time approximation. *Theoretical Computer Science* 41, 459–477 (2007)

- [CJ01] Cai, L., Juedes, D.W.: Subexponential Parameterized Algorithms Collapse the W -Hierarchy. In: Yu, Y., Spirakis, P.G., van Leeuwen, J. (eds.) ICALP 2001. LNCS, vol. 2076, pp. 273–284. Springer, Heidelberg (2001)
- [CJ03] Cai, L., Juedes, D.W.: On the existence of subexponential parameterized algorithms. *Journal of Computing and Systems Science* 67, 789–807 (2003)
- [CDRST03] Cheetham, J., Dehne, F., Rau-Chaplin, A., Stege, U., Taillon, P.J.: Solving Large FPT Problems on Coarse Grained Parallel Machines. *Journal of Computer and Systems Sciences* 67(4), 691–706 (2003)
- [CK00] Chekuri, C., Khanna, S.: A PTAS for the Multiple Knapsack Problem. In: Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA 2000), pp. 213–222 (2000)
- [CCFHJKX05] Chen, J., Chor, B., Fellows, M., Huang, X., Juedes, D.W., Kanj, A., Xia, G.: Tight lower bounds for certain parameterized NP-hard problems. *Information and Computation* 201, 216–231 (2005)
- [CFM11] Chen, Y., Flum, J., Müller, M.: Lower Bounds for Kernelizations and Other Preprocessing Procedures. *Theory Comput. Syst.* 48(4), 803–839 (2011)
- [CK12] Chen, J., Kanj, I.: Parameterized Complexity and Subexponential-Time Computability. In: Bodlaender, H.L., et al. (eds.) *Fellows Festschrift*. LNCS, vol. 7370, pp. 162–195. Springer, Heidelberg (2012)
- [CM99] Chen, J., Miranda, A.: A Polynomial-Time Approximation Scheme for General Multiprocessor Scheduling. In: *Proc. ACM Symposium on Theory of Computing (STOC 1999)*, pp. 418–427. ACM Press (1999)
- [CKX10] Chen, J., Kanj, I.A., Xia, G.: Improved upper bounds for vertex cover. *Theor. Comput. Sci.* 411, 3736–3756 (2010)
- [CF12] Chen, J., Flum, J.: A Parameterized Halting Problem. In: Bodlaender, H.L., et al. (eds.) *Fellows Festschrift*. LNCS, vol. 7370, pp. 364–397. Springer, Heidelberg (2012)
- [CG07] Chen, J., Grohe, M.: An isomorphism between subexponential and parameterized complexity theory. *SIAM. J. Comput.* 37, 1228–1258 (2007)
- [Co87] Courcelle, B.: Recognizability and Second-Order Definability for Sets of Finite Graphs, Technical Report I-8634, Université de Bordeaux (1987)
- [CDF97] Courcelle, B., Downey, R.G., Fellows, M.: A Note on the Computability of Graph Minor Obstruction Sets for Monadic Second Order Ideals. *Journal of Universal Computer Science* 3, 1194–1198 (1997)
- [DeH08] Demaine, E., Hajiaghayi, M.: The Bidimensionality Theory and Its Algorithmic Applications. *Comput. J.* 51(3), 292–302 (2008)
- [Do03] Downey, R.G.: Parameterized complexity for the skeptic. In: 18th Annual Conference on Computational Complexity, pp. 147–169. IEEE (2003)
- [Do12] Downey, R.: The Birth and Early Years of Parameterized Complexity. In: Bodlaender, H.L., et al. (eds.) *Fellows Festschrift*. LNCS, vol. 7370, pp. 17–38. Springer, Heidelberg (2012)
- [DF92a] Fellows, M.R.: Fixed parameter tractability and completeness. *Congressus Numerantium* 87, 161–187 (1992)
- [DF92b] Downey, R.G., Fellows, M.: Fixed parameter intractability. In: *Proceedings of Seventh Annual Conference on Structure in Complexity*. IEEE Publication, pp. 36–50 (1992)
- [DF93] Downey, R.G., Fellows, M.: Fixed Parameter Tractability and Completeness III: Some Structural Aspects of the W -Hierarchy. In: Ambos-Spies, K., Homer, S., Schöning, U. (eds.) *Complexity Theory: Current Research*, pp. 166–191. Cambridge Univ. Press (1993)

- [DF95a] Downey, R.G., Fellows, M.R.: Fixed Parameter Tractability and Completeness I: Basic Theory. *SIAM Journal of Computing* 24, 873–921 (1995)
- [DF95b] Downey, R.G., Fellows, M.R.: Fixed Parameter Tractability and Completeness II: Completeness for $W[1]$. *Theoretical Computer Science A* 141, 109–131 (1995)
- [DF95c] Downey, R.G., Fellows, M.R.: Parametrized Computational Feasibility. In: Clote, P., Remmel, J. (eds.) *Feasible Mathematics II*, pp. 219–244. Birkhauser, Boston (1995)
- [DF98] Downey, R.G., Fellows, M.R.: *Parameterized Complexity*. Springer (1998)
- [DFta] Downey, R.G., Fellows, M.R.: *Fundamentals of Parameterized Complexity*. Springer (2012) (in preparation)
- [DFL08] Downey, R.G., Fellows, M., Langston, M.: Two special issue of *The Computer Journal* devoted to Parameterized Complexity 58(1,3) (2008)
- [DFKHW94] Downey, R.G., Fellows, M., Kapron, B., Hallett, M., Wareham, H.T.: The Parameterized Complexity of Some Problems in Logic and Linguistics. In: Matiyasevich, Y.V., Nerode, A. (eds.) *LFCS 1994. LNCS*, vol. 813, pp. 89–100. Springer, Heidelberg (1994)
- [DFMR08] Downey, R.G., Fellows, M., McCartin, C., Rosamond, F.: Parameterized approximation of dominating set problems. *Information Processing Letters* 109(1), 68–70 (2008)
- [DFR98] Downey, R.G., Fellows, M., Regan, K.: Parameterized circuit complexity and the W -hierarchy. *Theoretical Computer Science* 191, 97–115 (1998)
- [DFS98] Downey, R.G., Fellows, M., Stege, U.: *Parameterized Complexity: A Framework for Systematically Confronting Computational Intractability*. DIMACS series on Combinatorics in the 21st Century. AMS Publ. (1998)
- [DFGW07] Downey, R., Flum, J., Grohe, M., Weyer, M.: Bounded fixed-parameter tractability and reducibility. *Annals of Pure and Applied Logic* 148, 1–19 (2007)
- [DTH11] Downey, R.G., Thilikos, D.: Confronting intractability via parameters. *Computer Science Review* 5, 279–317 (2011)
- [ELRW11] Eblen, J.D., Langston, M.A., Rogers, G.L., Weerapurage, D.P.: Parallel Vertex Cover: A Case Study in Dynamic Load Balancing. In: *Proceedings of Australasian Symposium on Parallel and Distributed Computing*, Perth, Australia (January 2011)
- [Ed65] Edmonds, J.: Paths, trees and flowers. *Canadian J. Math.* 17, 449–467 (1965)
- [EGG08] Eickmeyer, K., Grohe, M., Grüber, M.: Approximation of Natural $W[P]$ -Complete Minimisation Problems Is Hard. In: *IEEE Conference on Computational Complexity*, pp. 8–18 (2008)
- [EJS01] Erlebach, T., Jansen, K., Seidel, E.: Polynomial Time Approximation Schemes for Geometric Graphs. In: *Proc. ACM Symposium on Discrete Algorithms (SODA 2001)*, pp. 671–679 (2001)
- [FL87] Fellows, M., Langston, M.: Nonconstructive Proofs of Polynomial-Time Complexity. *Information Processing Letters* 26, 157–162 (1987/1988)
- [FL88] Fellows, M., Langston, M.: Nonconstructive Tools for Proving Polynomial-Time Complexity. *Journal of the Association for Computing Machinery* 35, 727–739 (1988)
- [FL89] Fellows, M.R., Langston, M.A.: An Analogue of the Myhill-Nerode Theorem and its Use in Computing Finite-Basis Characterizations. In: *Proceedings of the IEEE Symposium on the Foundations of Computer Science*, pp. 520–525 (1989)
- [FRRS06] Fellows, M., Rosamond, F., Rotics, U., Szeider, S.: Cliquewidth minimization is NP-hard. In: *Proceedings STOC 2006*, pp. 354–362 (2006)
- [FRRS09] Fellows, M., Rosamond, F., Rotics, U., Szeider, S.: Cliquewidth NP-complete. *SIAM J. on Discrete Mathematics* 23, 909–939 (2009)

- [FG02a] Flum, J., Grohe, M.: Describing Parameterized Complexity Classes. In: Alt, H., Ferreira, A. (eds.) STACS 2002. LNCS, vol. 2285, pp. 359–371. Springer, Heidelberg (2002)
- [FG02b] Flum, J., Grohe, M.: The Parameterized Complexity of Counting Problems. In: Conference version appeared in Proceedings of the 43rd IEEE Symposium on Foundations of Computer Science (FOCS 2002), pp. 538–547 (2002)
- [FG04] Flum, J., Grohe, M.: Parameterized Complexity and Subexponential Time. Bulletin of the European Association for Theoretical Computer Science 84 (October 2004)
- [FG06] Flum, J., Grohe, M.: Parameterized Complexity Theory. Springer (2006)
- [deF70] de Fluiter, B.: Algorithms for graphs of small treewidth, PhD Thesis, Utrecht Univ. (1970)
- [FGLS10] Fomin, F., Golovach, P., Lokshtanov, D., Saurabh, S.: Intractability of Clique-Width Parameterizations. SIAM J. Comput. 39(5), 1941–1956 (2010)
- [FS11] Fortnow, L., Santhanam, R.: Infeasible instances of compression and succinct pep’s. Journal of Computing and System Sciences 77, 91–106 (2011)
- [FrG01] Frick, M., Grohe, M.: Deciding First Order Properties of Locally Tree-Decomposable Structures. J. ACM 48, 1184–1206 (2001)
- [FrG02] Frick, M., Grohe, M.: The Complexity of First-Order and Monadic Second-Order Logic Revisited. In: LICS, pp. 215–224 (2002)
- [Fou03] Fouhy, J.: Computational Experiments on Graph Width Metrics, MSc Thesis, Victoria University, Wellington (2003)
- [GJ79] Garey, M., Johnson, D.: Computers and Intractability: A Guide to the Theory of NP-completeness. W.H. Freeman, San Francisco (1979)
- [GS12] Gaspers, S., Szeider, S.: Backdoors to Satisfaction. In: Bodlaender, H.L., et al. (eds.) Fellows Festschrift. LNCS, vol. 7370, pp. 287–317. Springer, Heidelberg (2012)
- [GGKS95] Goldberg, P., Golumbic, M., Kaplan, H., Shamir, R.: Four Strikes Against DNA Physical mapping. Journal of Computational Biology 2(1), 139–152 (1995)
- [GNR01] Gramm, J., Niedermeier, R., Rossmann, P.: Exact Solutions for Closest String and Related Problems. In: Eades, P., Takaoka, T. (eds.) ISAAC 2001. LNCS, vol. 2223, pp. 441–453. Springer, Heidelberg (2001)
- [Gr01a] Grohe, M.: Generalized Model-Checking Problems for First-Order Logic. In: Ferreira, A., Reichel, H. (eds.) STACS 2001. LNCS, vol. 2010, pp. 12–26. Springer, Heidelberg (2001)
- [GK11] Grohe, M., Kreutzer, S.: Methods for Algorithmic Meta Theorems. In: Grohe, M., Makowsky, J. (eds.) Model Theoretic Methods in Finite Combinatorics. Contemporary Mathematics, vol. 558. American Mathematical Society (2011)
- [GMKW11] Grohe, M., Marx, D., Kawarabayashi, K., Wollan, P.: Finding Topological Subgraphs is Fixed-Parameter Tractable. In: Proceedings of the 43rd ACM Symposium on Theory of Computing (STOC 2011), pp. 479–488 (2011)
- [GY12] Gutin, G., Yeo, A.: Constraint Satisfaction Problems Parameterized Above or Below Tight Bounds: A Survey. In: Bodlaender, H.L., et al. (eds.) Fellows Festschrift. LNCS, vol. 7370, pp. 257–286. Springer, Heidelberg (2012)
- [HM07] Hallett, M., McCartin, C.: A Faster FPT Algorithm for the Maximum Agreement Forest Problem. Theory of Computing Systems 41(3) (2007)
- [HKSWWta] Hermelin, D., Kratsch, S., Soltys, K., Whalstrom, M., Wu, X.: Hierarchies of inefficient kernelization (to appear)
- [Hi52] Higman, G.: Ordering by divisibility in abstract algebras. Proc. London Math. Soc. 2, 326–336 (1952)

- [HW06] Hlineny, P., Whittle, G.: Matroid tree-width. *Eur. J. Comb.* 27(7), 1117–1128 (2006)
- [IPZ01] Impagliazzo, R., Paturi, R., Zane, F.: Which problems have strongly exponential complexity? *JCSS* 63(4), 512–530 (2001)
- [KW90] Kannan, S., Warnow, T.: Inferring Evolutionary History from DNA Sequences. In: *Proceedings of the 31st Annual Symposium on the Theory of Computing*, pp. 362–378 (1990)
- [KST94] Kaplan, H., Shamir, R., Tarjan, R.E.: Tractability of Parameterized Completion Problems on Chordal and Interval Graphs: Minimum Fill-In and DNA Physical Mapping. In: *Proc. 35th Annual Symposium on the Foundations of Computer Science (FOCS)*, pp. 780–791. IEEE Press (1994)
- [Ka72] Karp, R.: Reducibility Among Combinatorial Problems. In: *Complexity of Computer Computations*, pp. 85–103 (1972)
- [Ka11] Karp, R.: Heuristic algorithms in computational molecular biology. *J. Comput. Syst. Sci.* 77(1), 122–128 (2011)
- [KR02] Khot, S., Raman, V.: Parameterized Complexity of Finding Subgraphs with Hereditary properties. *Theoretical Computer Science* 289, 997–1008 (2002)
- [Ko12] Koblitz, N.: Crypto Galore! In: Bodlaender, H.L., Downey, R., Fomin, F.V., Marx, D. (eds.) *Fellows Festschrift. LNCS*, vol. 7370, pp. 39–50. Springer, Heidelberg (2012)
- [Ku30] Kuratowski, K.: Sur le probleme des courbes gauches en topologie. *Fund. Math.* 15, 271–283 (1930)
- [La12] Langston, M.: Fixed-Parameterized Tractability, a Prehistory. In: Bodlaender, H.L., Downey, R., Fomin, F.V., Marx, D. (eds.) *Fellows Festschrift. LNCS*, vol. 7370, pp. 3–16. Springer, Heidelberg (2012)
- [LPSSV08] Langston, M., Perkins, A., Saxton, A., Scharff, J., Voy, B.: Innovative Computational Methods for Transcriptomic Data Analysis: A Case Study in the Use of FPT for Practical Algorithm Design and Implementation. *The Computer Journal* 51, 26–38 (2008)
- [Le83] Lenstra, H.: Integer Programming with a Fixed Number of Variables. *Mathematics of Operations Research* 8, 538–548 (1983)
- [LMS12] Lokshtanov, D., Misra, N., Saurabh, S.: Kernelization – Preprocessing with A Guarantee. In: Bodlaender, H.L., Downey, R., Fomin, F.V., Marx, D. (eds.) *Fellows Festschrift. LNCS*, vol. 7370, pp. 129–161. Springer, Heidelberg (2012)
- [MR99] Mahajan, M., Raman, V.: Parameterizing Above Guaranteed Values: MaxSat and MaxCut. *J. Algorithms* 31, 335–354 (1999)
- [Ma12] Marx, D.: What’s Next? Future Directions in Parameterized Complexity. In: Bodlaender, H.L., Downey, R., Fomin, F.V., Marx, D. (eds.) *Fellows Festschrift. LNCS*, vol. 7370, pp. 469–496. Springer, Heidelberg (2012)
- [McC06] McCartin, C.: Parameterized counting problems. *Annals of Pure and Applied Logic* 138, 147–182 (2006)
- [Mo99] Mohar, B.: A Linear Time Algorithm for Embedding Graphs in an Arbitrary Surface. *SIAM J. Discrete Math.* 12, 6–26 (1999)
- [Mu06] Müller, M.: Randomized Approximations of Parameterized Counting Problems. In: Bodlaender, H.L., Langston, M.A. (eds.) *IWPEC 2006. LNCS*, vol. 4169, pp. 50–59. Springer, Heidelberg (2006)
- [Mu08] Müller, M.: Valiant-Vazirani Lemmata for Various Logics. *Electronic Colloquium on Computational Complexity (ECCC)* 15(063) (2008)
- [Nie02] Niedermeier, R.: Invitation to Fixed-Parameter Algorithms. *Habilitationsschrift, University of Tübingen* (September 2002)

- [Nie06] Niedermeier, R.: Invitation to Fixed-Parameter Algorithms. Oxford University Press (2006)
- [NR00] Niedermeier, R., Rossmanith, P.: A general method to speed up fixed-parameter tractable algorithms. *Information Processing Letters* 73, 125–129 (2000)
- [NT75] Nemhauser, G., Trotter Jr., L.: Vertex packings: Structural properties and algorithms. *Mathematical Programming* 8, 232–248 (1975)
- [PY97] Papadimitriou, C., Yannakakis, M.: On the Complexity of Database Queries. In: *Proc. ACM Symp. on Principles of Database Systems*, pp. 12–19 (1997); Journal version in *Journal of Computer System Sciences* 58, 407–427 (1999)
- [ST98] Shamir, R., Tzur, D.: The Maximum Subforest Problem: Approximation and Exact Algorithms. In: *Proc. ACM Symposium on Discrete Algorithms*, pp. 394–399. ACM Press (1998)
- [St00] Stege, U.: Resolving Conflicts in Problems in Computational Biochemistry. Ph.D. dissertation, ETH (2000)
- [St12] Stege, U.: The Impact of Parameterized Complexity to Interdisciplinary Problem Solving. In: Bodlaender, H.L., Downey, R., Fomin, F.V., Marx, D. (eds.) *Fellows Festschrift. LNCS*, vol. 7370, pp. 56–68. Springer, Heidelberg (2012)
- [RSV04] Reed, B., Smith, K., Vetta, A.: Finding odd cycle transversals. *Operations Research Letters* 32, 299–301 (2004)
- [RS86a] Robertson, N., Seymour, P.D.: Graph minors. II. Algorithmic aspects of tree-width. *Journal of Algorithms* 7, 309–322 (1986)
- [T12] Thilikos, D.: Graph Minors and Parameterized Algorithm Design. In: Bodlaender, H.L., Downey, R., Fomin, F.V., Marx, D. (eds.) *Fellows Festschrift. LNCS*, vol. 7370, pp. 228–256. Springer, Heidelberg (2012)
- [Wa37] Wagner, K.: Über einer eigenschaft der eberner complexe. *Math. Ann.* 14, 570–590 (1937)

Kernelization – Preprocessing with a Guarantee

(For the 60th Birthday of Prof. Mike Fellows)

Daniel Lokshtanov¹, Neeldhara Misra², and Saket Saurabh²

¹ University of California, San Diego, USA
daniello@ii.uib.no

² The Institute of Mathematical Sciences, Chennai, India
{neeldhara,saket}@imsc.res.in

Abstract. Data reduction techniques are widely applied to deal with computationally hard problems in real world applications. It has been a long-standing challenge to formally express the efficiency and accuracy of these “pre-processing” procedures. The framework of parameterized complexity turns out to be particularly suitable for a mathematical analysis of pre-processing heuristics. A kernelization algorithm is a pre-processing algorithm which simplifies the instances given as input in polynomial time, and the extent of simplification desired is quantified with the help of the additional parameter.

We give an overview of some of the early work in the area and also survey newer techniques that have emerged in the design and analysis of kernelization algorithms. We also outline the framework of Bodlaender et al. [9] and Fortnow and Santhanam [38] for showing kernelization lower bounds under reasonable assumptions from classical complexity theory, and highlight some of the recent results that strengthen and generalize this framework.

1 Introduction

Preprocessing (data reduction or kernelization) as a strategy of coping with hard problems is universally used in almost every implementation. The history of preprocessing, such as applying reduction rules to simplify truth functions, can be traced back to the 1950’s [58]. A natural question in this regard is how to measure the quality of preprocessing rules proposed for a specific problem. For a long time the mathematical analysis of polynomial time preprocessing algorithms was neglected. A possible explanation for this phenomenon is that if we start with an instance I of an NP-hard problem and can show that in polynomial time we can replace this with an equivalent instance I' with $|I'| < |I|$ then that would imply $P=NP$. This makes it difficult to design the right definitions of efficient preprocessing within classical complexity. The situation changed drastically with advent of parameterized complexity [26]. Combining tools from parameterized complexity and classical complexity it has become possible to derive upper and lower bounds on the sizes of reduced instances, or so called *kernels*. The importance of preprocessing and the mathematical challenges it poses is beautifully expressed in the following quote by Fellows [30]:

It has become clear, however, that far from being trivial and uninteresting, that pre-processing has unexpected practical power for real world input distributions, and is mathematically a much deeper subject than has generally been understood.

Historically, the study of kernelization is rooted in parameterized complexity but it appeared soon that the challenges of kernelization tractability are deeply linked to classical polynomial time tractability. In the classical computational complexity originated from 1970s, we distinguish between tractable computational problems and intractable. This theory classifies problems according to how much time or space is required by algorithms to solve these problems, as a function of the size of the input. The tractability border is drawn at polynomial time solvability - a problem which has a polynomial time algorithm is considered tractable, while one that does not is considered intractable. However, ignoring the structural information about the input and defining intractability based only on input size can make some problems appear harder than they really are. Parameterized complexity attempts to address this issue by measuring computational resources such as time and space in terms of input size and additional parameters. In parameterized complexity the central notion of efficiency is “fixed parameter tractability”. The notion may be thought of as a generalization of polynomial time to a multivariate setting. The running time of a fixed parameter tractable algorithm is polynomial in the size of the input but can be exponential in terms of parameters. A surprisingly large number of intractable problems have been shown to exhibit fixed parameter tractable algorithms. A kernelization algorithm is a polynomial time algorithm reducing instances of parameterized problems to equivalent instances whose size can be upper bounded by a function of the parameter. Thus kernelization can be seen as a refinement of the notion of the classical polynomial time tractability from a parameterized perspective. The development of kernelization algorithms demonstrate the importance of the second (and maybe even other) measures and indicate that polynomial time computation is much more powerful than previously suggested.

Informally, in parameterized complexity each problem instance comes with a parameter k . As a warm-up, let us consider a few examples of parameterized problems. Our first example is about vertex cover. A set of vertices S in a graph is a vertex cover if every edge of the graph contains at least one vertex from S . In the parameterized vertex cover problem, we call it p -VERTEX COVER the parameter is an integer k and we ask whether the input graph has a vertex cover of size at most k . We will use p - to emphasise that we are considering a parameterized problem, rather than its classical counterpart. Another problem, p -LONGEST PATH asks whether a given graph contains a path of length at least k . And finally, p -DOMINATING SET is to decide whether a given graph has a dominating set of size k , that is, a set of vertices such that every vertex of the input graph is either in this set or is adjacent to some vertex from the set.

The parameterized problem is said to admit a *kernel* if there is a polynomial time algorithm (the degree of polynomial is independent of k), called a *kernelization* algorithm, that reduces the input instance down to an instance with

size bounded by some function $h(k)$ of k only, while preserving the answer. If the function $h(k)$ is polynomial in k , then we say that the problem admits a polynomial kernel.

In our examples, p -VERTEX COVER admits a polynomial kernel—there is a polynomial time algorithm that for any instance (G, k) of the problem outputs a new instance (G', k') such that G' has at most $2k$ vertices and G has a vertex cover at most k if and only if G' has a vertex cover of size at most k' [17]. The second example, p -LONGEST PATH, admits a kernel but the bounding function $h(k)$ is exponential. It is possible to show that up to some assumptions from complexity theory, the problem does not admit a polynomial kernel [9], even if the input graph G is required to be planar. Our last example, p -DOMINATING SET admits no kernel unless $\text{FPT}=\text{W}[2]$ yielding a collapse of several levels in the parameterized complexity hierarchy [26]. However, on planar graph p -DOMINATING SET admits a kernel of size $h(k) = \mathcal{O}(k)$, i.e. a linear kernel.

In this survey we discuss some of the classical and recent algorithmic techniques for obtaining kernels, and discuss some of the recent developments in deriving lower bounds on the sizes of the kernels. We do not try to give a comprehensive overview of all significant results in the area—doing this will require at least a book. Our objective is simply to give a glimpse into the exciting world of kernelization [27,52,6,36,12,38,9,13,22,50,11,20]. We refer to the surveys of Fellows [30] and Guo and Niedermeier [30,41] for further reading on kernelization algorithms. For a more detailed survey of kernelization lower bounds we refer to survey of Misra et al. [55].

2 Basic Definitions

A *parameterized problem* is a language $L \subseteq \Sigma^* \times \mathbb{N}$, where Σ is a finite alphabet and \mathbb{N} is the set of non-negative integers. The second component is called the *parameter* of the problem. The central notion in parameterized complexity is that of *fixed-parameter tractability*, which means given an instance (x, k) of a parameterized language L , deciding whether $(x, k) \in L$ in time $f(k) \cdot p(|x|)$, where f is an arbitrary function of k alone and p is a polynomial function. Such an algorithm is called a fixed-parameter tractable algorithm and we call a problem that admits an algorithm of this kind fixed-parameter tractable (FPT).

We now turn to the formal notion that captures the notion of simplification, which is what most heuristics do when applied to a problem. A *data reduction rule* for a parameterized language L is a function $\phi : \Sigma^* \times \mathbb{N} \rightarrow \Sigma^* \times \mathbb{N}$ that maps an instance (x, k) of L to an equivalent instance (x', k') of L such that

1. ϕ is computable in time polynomial in $|x|$ and k ;
2. $|x'| \leq |x|$.

Two instances of L are equivalent if $(x, k) \in L$ if and only if $(x', k') \in L$.

In general, a *kernelization algorithm* consists of a finite set of data reduction rules such that by applying the rules to an instance (x, k) (in some specified order) one obtains an instance (x', k') with the property that $|x'| \leq g(k)$ and

$k' \leq g(k)$, for some function g only depending on k . Such a “reduced” instance is called a *problem kernel* and $g(k)$ is called the *kernel size*. Formally, this is defined as follows.

Definition 1. [Kernelization, Kernel] [9] *A kernelization algorithm for a parameterized problem $\Pi \subseteq \Sigma^* \times \mathbb{N}$ is an algorithm that, given $(x, k) \in \Sigma^* \times \mathbb{N}$, outputs, in time polynomial in $(|x| + k)$, a pair $(x', k') \in \Sigma^* \times \mathbb{N}$ such that: (a) $(x, k) \in \Pi$ if and only if $(x', k') \in \Pi$ and (b) $|x'|, k' \leq g(k)$, where g is some computable function. The output instance x' is called the kernel, and the function g is referred to as the size of the kernel. If $g(k) = k^{O(1)}$, then we say that Π admits a polynomial kernel.*

It is important to mention here that the early definitions of kernelization required that $k' \leq k$. On an intuitive level this makes sense, as the parameter k measures the complexity of the problem — thus the larger the k , the harder the problem. This requirement was subsequently relaxed, notably in the contest of lower bounds. An advantage of the more liberal notion of kernelization is that it is robust with respect to polynomial transformations of the kernel. However, it limits the connection with practical preprocessing. All the kernels obtained in this paper respect the fact that the output parameter is at most the input parameter, that is, $k' \leq k$.

The following lemma tells us that a parameterized problem Π is in FPT if and only if there exists a computable function g such that Π admits a kernel of size $g(k)$.

Lemma 1 ([27]). *If a parameterized problem Q is FPT via a computable function then it admits kernelization.*

Proof. Suppose that there is an algorithm deciding if $x \in Q$ in time $f(k)|x|^c$ time for some computable function f and constant c . If $|x| \geq f(k)$, then we run the decision algorithm on the instance in time $f(k)|x|^c \leq |x|^{c+1}$. If the decision algorithm outputs YES, the kernelization algorithm outputs a constant size YES instance, and if the decision algorithm outputs NO, the kernelization algorithm outputs a constant size NO instance. On the other hand, if $|x| < f(k)$, then the kernelization algorithm outputs x . This yields a kernel of size $f(k)$ for the problem. \square

However, kernels obtained by this theoretical result are usually of exponential (or even worse) size, while problem-specific data reduction rules often achieve quadratic ($g(k) = O(k^2)$) or even linear-size ($g(k) = O(k)$) kernels. So a natural question for any concrete FPT problem is whether it admits polynomial-time kernelization to a problem kernel that is bounded by a polynomial function of the parameter ($g(k) = O(k^{O(1)})$).

Polynomial kernels form our basic notion of efficient kernelization. For a comprehensive study of fixed-parameter tractability and kernelization, we refer to the books [26,34,56] and the surveys [41,55].

Notations. We conclude this section with some graph-theoretic notations. We follow the style of [24]. Let $G = (V, E)$ be a graph. For a vertex v in G , we write $N_G(v)$ to denote the set of v 's neighbors in G , and we write $d_G(v)$ to denote the *degree* of v , that is, the number of v 's neighbors in G . If it is clear from the context which graph is meant, we write $N(v)$ and $d(v)$, respectively, for short. A graph $G' = (V', E')$ is a *subgraph* of G if $V' \subseteq V$ and $E' \subseteq E$. The subgraph G' is called an *induced subgraph* of G if $E' = \{\{u, v\} \in E \mid u, v \in V'\}$, in this case, G' is also called the subgraph *induced by* V' and denoted with $G[V']$. A vertex v *dominates* a vertex u if $u \in N(v)$.

3 Classical Techniques Explained via Simple Examples

In this section we give several examples of techniques used to obtain kernels, often polynomial kernels. Some of them are almost trivial and some of them are more involved. We start with the parameterized version of MAX-3-SAT. Our other examples in this section include a polynomial kernel for p -FEEDBACK ARC SET IN TOURNAMENTS (p -FAST), d -HITTING SET using the *Sunflower Lemma*, kernels for p -DUAL VERTEX COLORING and p -MAX-SAT using *crown decomposition* and an exponential kernel for p -EDGE CLIQUE COVER.

3.1 Max-3-Sat

Let F be a given boolean CNF 3-SAT formula with n variables and m clauses. In the optimization version of the problem the task is to find a truth assignment satisfying the maximum number of clauses. The parameterized version of the problem is the following.

p -MAX-3-SAT

Instance: A 3-CNF formula F , and a non-negative integer k .

Parameter: k .

Problem: Decide whether F has a truth assignment satisfying at least k clauses.

Let (F, k) be an instance of p -MAX-3-SAT and let m be the number of clauses in F and n the number of variables. It is well known that in any boolean CNF formula, there is an assignment that satisfies at least half of the clauses (given any assignment that does not satisfy half the clauses, its bitwise complement will). So if the parameter k is less than $m/2$, then there is always an assignment to the variables that satisfies at least k of the clauses. In this case, we reduce the instance to the trivial instance with one clause and the parameter $k = 1$, which is always a YES instance. Otherwise, $m \leq 2k$. By deleting all variables that do not occur in any clause we obtain that $n \leq 6k$, implying that the input instance is a kernel of polynomial size..

3.2 Kernelization for FAST

In this section we discuss a kernel for p -FAST. A *tournament* is a directed graph T such that for every pair of vertices $v, u \in V(T)$, either uv or vu is an arc of T . A set of arcs A of T is called a *feedback arc set*, if every cycle of T contains an arc from A . In other words, removal of A from T turns it into an acyclic graph.

p -FAST

Instance: A tournament T and a non-negative integer k .

Parameter: k .

Problem: Decide whether T has a feedback arc set of size at most k .

Lemma 2 ([25]). p -FAST admits a kernel with at most $k^2 + 2k$ vertices.

Proof. The following observation is useful. A graph is acyclic if and only if it is possible to order its vertices in such a way such that for every arc uv , we have $u < v$. Hence a set of arcs A is an inclusion minimal feedback arc set if and only if A is an inclusion minimal set such that reversing directions of all arcs from A results in an acyclic tournament.

In what follows by a *triangle* we mean a directed triangle. We give two simple reduction rules.

Rule 1. If an arc e is contained in at least $k + 1$ triangles, then reverse e and reduce k by 1.

Rule 2. If a vertex v is not contained in any triangle, then delete v from T .

Let us remark that after applying any of the two rules, the resulting graph is again a tournament.

The first rule is sound because if we do not reverse e , we have to reverse at least one arc from each of $k + 1$ triangles containing e . Thus e belongs to every feedback arc set of size at most k .

For the correctness of the second rule. Let X be the set of heads of arcs with tail in v and let Y be the sets of tails of arcs with head in v . Because T is a tournament, X and Y is a partition of $V(T) \setminus \{v\}$. Since v is not a part of any triangle in T , we have that there is no arc from X to Y . Moreover, for any feedback arc set A_1 of tournament $T[X]$ and any feedback arc set A_2 of tournament $T[Y]$, the set $A_1 \cup A_2$ is feedback arc set of T . Thus (T, k) is a YES instance if and only if $(T \setminus v, k)$ is.

Finally we show that any reduced YES instance T has at most $k(k+2)$ vertices. Let A be a feedback arc set of T of size at most k . For every arc $e \in A$, aside from the two endpoints of e , there are at most k vertices that are contained in a triangle containing e , because otherwise the first rule would have applied. Since every triangle in T contains an arc of A and every vertex of T is in a triangle, we have that T has at most $k(k + 2)$ vertices. \square

3.3 p -d-Hitting Set

Our next example is a kernelization for the p -d-HITTING SET problem, established in [1]. We follow the presentation in [34].

p - d -HITTING SET

Instance: A family \mathcal{F} of sets over an universe \mathcal{U} , each of cardinality d and a positive integer k

Parameter: k

Problem: Decide whether there is a subset $U \subseteq \mathcal{U}$ of size at most k such that U contains at least one element from each set in \mathcal{F} .

The kernelization algorithm is based on the following widely used Sunflower Lemma. We first define the terminology used in the statement of the lemma. A *sunflower* with k petals and a core Y is a collection of sets S_1, S_2, \dots, S_k such that $S_i \cap S_j = Y$ for all $i \neq j$; the sets $S_i \setminus Y$ are petals and we require none of them to be empty. Note that a family of pairwise disjoint sets is a sunflower (with an empty core). We need the following classical result of Erdős and Rado [29], see also [34].

Lemma 3 ([29,34]). [Sunflower Lemma] *Let \mathcal{F} be a family of sets over an universe \mathcal{U} each of cardinality d . If $|\mathcal{F}| > d!(k-1)^d$ then \mathcal{F} contains a sunflower with k petals and such a sunflower can be computed in time polynomial in the size of \mathcal{F} and \mathcal{U} .*

Now we are ready to prove the following theorem about kernelization for p - d -HITTING SET

Theorem 1 ([34]). *p - d -HITTING SET admits a kernel with $\mathcal{O}(k^d \cdot d!)$ sets and $\mathcal{O}(k^d \cdot d! \cdot d)$ elements.*

Proof. The crucial observation is that if \mathcal{F} contains a sunflower $S = \{S_1, \dots, S_{k+1}\}$ of cardinality $k+1$ then every hitting set H of \mathcal{F} of cardinality k must intersect with the core Y of the sunflower S . Indeed, if H does not intersect C , it should intersect each of the $k+1$ disjoint petals $S_i \setminus C$. Therefore if we let $\mathcal{F}' = (\mathcal{F} \setminus S) \cup Y$, then the instances $(\mathcal{U}, \mathcal{F}, k)$ and $(\mathcal{U}, \mathcal{F}', k)$ are equivalent.

Now we apply the Sunflower Lemma for all $d' \in \{1, \dots, d\}$ on collections of sets with d' elements by repeatedly replacing sunflowers of size at least $k+1$ with their cores until the number of sets for any fixed $d' \in \{1, \dots, d\}$ is at most $\mathcal{O}(k^{d'} d')$. We also remove elements which do not belong to any set. Summing over all d' , we obtain that the new family of sets \mathcal{F}' contains $\mathcal{O}(k^d \cdot d!)$ sets. Every set contains at most d elements, and thus the amount of elements in the kernel is $\mathcal{O}(k^d \cdot d! \cdot d)$. \square

3.4 Kernels via Crown Decomposition

Crown decomposition is a general kernelization technique that can be used to obtain kernels for many problems. The technique is based on the classical matching theorems of König and Hall [44,49].

Definition 2. *A crown decomposition of a graph $G = (V, E)$ is a partitioning of V as C, H and R , where C and H are nonempty and the partition satisfies the following properties.*

1. C is an independent set.
2. There are no edges between vertices of C and R , i.e. H separates C and R ;
3. Let E' be the set of edges between vertices of C and H . Then E' contains a matching of size $|H|$.

Set C can be seen as a crown put on head H of the remaining part R of the Royal body. Fig. 1 provides an example of a crown decomposition. Let us remark that the fact that E' contains a matching of size $|H|$ implies that there is a matching of H into C , i. e. a matching in the bipartite subgraph $G' = (C \cup H, E')$ saturating all the vertices of H .

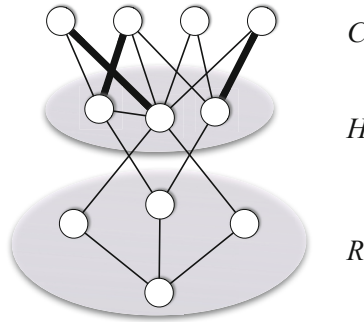


Fig. 1. Example of a crown decomposition. Set C is an independent set, H separates C and R , and H has a matching into C .

The following lemma, which establishes that crown decompositions can be found in polynomial time, is the basis for kernelization algorithms using crown decompositions.

Lemma 4 ([18]). [Crown Lemma] *Let G be a graph without isolated vertices and with at least $3k+1$ vertices. There is a polynomial time algorithm that either*

- Find a matching of size $k+1$ in G ; or
- Find a crown decomposition of G .

We demonstrate the application of crown decompositions on kernelization for p -DUAL VERTEX COLORING and p -MAX-SAT.

Dual of Coloring. In this section we show how Crown Lemma can be used to obtain kernel for p -DUAL VERTEX COLORING. This problem concerns coloring of the vertex set of a graph. A k -coloring c of an undirected graph $G = (V, E)$ assigns a color to each vertex of the graph $c : V \rightarrow \{1, 2, \dots, k\}$ such that adjacent vertices have different colours. The smallest k for which G has a k -coloring is called the *chromatic number* of G , denoted by $\chi(G)$. It is well known that deciding if $\chi(G)$ is at most 3 is an NP-complete problem. Thus from the Parameterized Complexity perspective, the following parameterization is more interesting.

p -DUAL VERTEX COLORING

Instance: A graph $G = (V, E)$ and a non-negative integer k .

Parameter: k

Problem: Decide whether G has a $(|V| - k)$ -coloring.

It is easier to apply Crown Decomposition to the complement of the input graph. The *complement* of undirected graph $G = (V, E)$ is denoted by \overline{G} ; its vertex set is V and its edge set is $\overline{E} = \{uv : uv \notin E, u \neq v\}$. coloring an n -vertex graph in $(n - k)$ colours is equivalent to covering its complement by $(n - k)$ cliques.

Given a crown decomposition (C, H, R) of \overline{G} , we apply the following Rule.

Crown Rule for Dual Vertex Coloring: Construct a new instance of the problem (G', k') by removing $H \cup C$ from G and reducing k by $|H|$. In other words, $G' = G[R]$ and $k' = k - |H|$.

The Rule is sound by the following lemma.

Lemma 5 ([18]). *Let (C, H, R) be a crown decomposition of \overline{G} . Then $(G = (V, E), k)$ is a YES instance if and only if $(G' = (V', E'), k')$ is a YES instance.*

Proof. We want to show that G is $(|V| - k)$ -colorable if and only if $G' = G[R]$ is $(|V'| - k')$ -colorable, where $k' = k - |H|$.

Let c be a $(|V| - k)$ -coloring of G . Because C is a clique, all vertices of C are assigned to different colours by c . None of these colours can be used on vertices of R because every vertex from R is adjacent to all vertices of C . Thus c uses on $G' = G[R]$ at most

$$|V| - k - |C| = |V| - (|C| + |H|) - (k - |H|) = |V'| - k'$$

colors.

Now let c' be a $(|V'| - k')$ -coloring of G' . We take $|C|$ new colors to color vertices of C . Because there is a matching M of H into C in \overline{G} , we can use the same $|C|$ colors that were used on C to color H . For every vertex $u \in H$, we select a color of the vertex from C matched by M to u . To color G , we used at most

$$|V'| - k' + |C| = |V| - (|C| + |H|) - (k - |H|) + |C| = |V| - k$$

colors. This completes the proof. \square

Theorem 2 ([18]). *p -DUAL VERTEX COLORING has a kernel with at most $3k - 3$ vertices.*

Proof. For an input n -vertex graph G and a positive integer k , we take the complement of G . If complement \overline{G} contains an isolated vertex v , then in G this vertex v is adjacent to all other vertices, and thus (G, k) is a YES instance if and only if $(G \setminus v, k - 1)$ is a YES instance.

Let us assume that \overline{G} has no isolated vertices. We apply Crown Lemma on \overline{G} . If \overline{G} has a matching M of size k , then G is $(n - k)$ -colorable. Indeed, the endpoints of every edge of M can be colored with the same color. If \overline{G} has no matching M of size k , then either $n \leq 3(k - 1)$, or G can be reduced by making use of the Crown Rule for p -DUAL VERTEX COLORING. \square

Maximum Satisfiability. Our next example concerns MAX-SAT. We are interested in the following parameterized version of MAX-SAT.

p -MAX-SAT

Instance: A CNF formula F , and a non-negative integer k .

Parameter: k

Problem: Decide whether F has a truth assignment satisfying at least k clauses.

Theorem 3 ([53]). p -MAX-SAT admits a kernel with at most k variables and $2k$ clauses.

Proof. Let F be a CNF formula with n variables and m clauses. If we assign values to the variables uniformly at random, linearity of expectation yields that the expected number of satisfied clauses is at least $m/2$. Since there has to be at least one assignment satisfying at least the expected number of clauses this means that if $m \geq 2k$ then (F, k) is a YES instance. In what follows we show how to give a kernel with $n < k$ variables. Whenever possible we apply a cleaning rule; if some variable does not occur in any clauses, remove the variable.

Let G_F be the *variable-clause* incidence graph of F . That is, G_F is a bipartite graph with bipartition (X, Y) . The set X corresponds to the variables of F and Y corresponds to the clauses. For a vertex $x \in X$ we will refer to x as both the vertex in G_F and the corresponding variable in F . Similarly, for a vertex $c \in Y$ we will refer to c as both the vertex in G_F and the corresponding clause in F . In G_F there is an edge between a variable $x \in X$ and a clause $c \in Y$ if and only if either x , or its negation is in c . If there is a matching of X into Y in G_F , then there is a truth assignment satisfying at least $|X|$ clauses. This is true because we can set each variable in X in such a way that the clause matched to it becomes satisfied. Thus at least $|X|$ clauses are satisfied. Hence, in this case if $k \leq |X|$ then (F, k) is a YES instance. We now show that if F has at least k variables, then we can in polynomial time, either reduce F to an equivalent smaller instance or find an assignment to the variables satisfying at least k clauses.

Suppose F has at least k variables. Using Hall's Theorem and a polynomial time algorithm computing maximum-size matching, we can in polynomial time find either a matching of X into Y or an inclusion minimal set $C \subseteq X$ such that $|N(C)| < |C|$. If we found a matching we are done, as we can satisfy at least $|X| \geq k$ clauses. So suppose we found a set C as described. Let H be $N(C)$ and $R = V(G_F) \setminus (C \cup H)$. Clearly, $N(C) \subseteq H$, $N(R) \subseteq H$ and $G[C]$ is an independent set. Furthermore, for a vertex $x \in C$ we have that there is a matching of $C \setminus x$ into H since $|N(C')| \geq |C'|$ for every $C' \subseteq C \setminus x$. Since $|C| > |H|$, we have that the matching from $C \setminus x$ to H is in fact a matching of H into C . Hence (C, H, R) is a crown decomposition of G_F .

We prove that all clauses in H are satisfied in every truth assignment to the variables satisfying the maximum number of clauses. Indeed, consider any truth assignment t that does not satisfy all clauses in H . For every variable y in $C \setminus \{x\}$

change the value of y such that the clause in H matched to y is satisfied. Let t' be the new assignment obtained from t in this manner. Since $N(C) \subseteq H$ and t' satisfies all clauses in H , more clauses are satisfied by t' than by t . Hence t can not be an assignment satisfying the maximum number of clauses.

The argument above shows that (F, k) is a YES instance to p -MAX-SAT if and only if $(F \setminus (C \cup H), k - |H|)$ is. This gives rise to a simple reduction rule: remove $(C \cup H)$ from F and decrease k by $|H|$. This completes the proof of the theorem. \square

3.5 Clique Cover

Unfortunately, not all known problem kernels are shown to have polynomial size. Here, we present the example of p -EDGE CLIQUE COVER, and the reduction rules presented here lead to an exponential-size kernel. It has been a pressing challenge for a long time to find out if this can be improved to a polynomial sized kernel. In recent news, the answer to this question has been announced to be in the negative, that is to say, the problem is unlikely to admit a polynomial kernel under reasonable complexity-theoretic assumptions [20]. The problem is the following.

p -EDGE CLIQUE COVER

Instance: A graph $G = (V, E)$, and a non-negative integer k .

Parameter: k

Problem: Decide whether edges of G can be covered by at most k cliques.

We use $N(v)$ to denote the neighborhood of vertex v in G , namely, $N(v) := \{u \mid uv \in E\}$. The *closed* neighborhood of vertex v , denoted by $N[v]$, is $N(v) \cup \{v\}$. We describe data reduction rules for a generalized version of p -EDGE CLIQUE COVER, in which already some edges may be marked as “covered”. Then, the question is to find a clique cover of size k that covers all uncovered edges. We apply the following data reduction rules from [40]:

Rule 1. Remove isolated vertices and vertices that are only adjacent to covered edges.

Rule 2. If there is an edge uv whose endpoints have exactly the same closed neighborhood, that is, $N[u] = N[v]$, then mark all edges incident to u as covered. To reconstruct a solution for the non-reduced instance, add u to every clique containing v .

Theorem 4 ([40]). p -EDGE CLIQUE COVER admits a kernel with at most 2^k vertices.

Proof. Let $G = (V, E)$ be a graph that has a clique cover C_1, \dots, C_k and such that none of two Rules can be applied to G . We claim that G has at most 2^k vertices. Targeting towards a contradiction, let us assume that G has more than 2^k vertices. We assign to each vertex $v \in V$ a binary vector b_v of length k where bit i , $1 \leq i \leq k$, is set to 1 if and only if v is contained in clique C_i . Since there are

only 2^k possible vectors, there must be $u \neq v \in V$ with $b_u = b_v$. If b_u and b_v are zero vectors, the first rule applies; otherwise, u and v are contained in the same cliques. This means that u and v are adjacent and share the same neighborhood, and thus the second rule applies. Hence, if G has more than 2^k vertices, at least one of the reduction rules can be applied to it, which is a contradiction to the initial assumption. \square

4 Recent Upper Bound Machinery

In this section we survey recent methods to obtain polynomial kernels. This includes reduction rules based on protrusions, probabilistic methods and matroids.

4.1 Protrusion Based Replacement

In this part we discuss kernelization for different classes of sparse graphs. An important result in the area of kernelization is by Alber et al. [2]. They obtained a linear sized kernel for the p -DOMINATING SET problem on planar graphs. This work triggered an explosion of papers on kernelization, and in particular on kernelization of problems on planar and different classes of sparse graphs. Combining the ideas of Alber et al. with *problem specific* data reduction rules, linear kernels were obtained for a variety of parameterized problems on planar graphs including p -CONNECTED VERTEX COVER, p -INDUCED MATCHING and p -FEEDBACK VERTEX SET. In 2009 Bodlaender et al. [7] obtained meta kernelization algorithms that eliminated the need for the design of problem specific reduction rules by providing an automated process that generates them. They show that all problems that have a “distance property” and are expressible in a certain kind of logic or “behave like a regular language” admit a polynomial kernel on graphs of bounded genus. In what follows we give a short description of these meta theorems.

Informal Description. The notion of “protrusions and finite integer index” is central to recent meta kernelization theorems. In the context of problems on graphs, there are three central ideas that form the undercurrent of all protrusion-based reduction rules:

- describing an equivalence that classifies all instances of a problem in an useful manner,
- the ability to easily identify, given a problem, whether the said equivalence has finite index,
- given an instance of a problem, finding large subgraphs that “can be replaced” with smaller subgraphs that are equivalent to the original.

One of the critical aspects of this development is coming up with the right definition for describing the circumstances in which a subgraph may be replaced. This is captured by the notion of a protrusion.

In general, an r -*protrusion* in a graph G is simply a subgraph $H = (V_H, E_H)$ such that the number of vertices in H that have neighbours in $G \setminus H$ is at most r and the treewidth of H is at most r . The *size* of the protrusion is the number of vertices in it, that is, $|V_H|$. The vertices in H that have neighbours in $G \setminus H$ comprise the *boundary* of H . Informally, H may be thought of as a part of the graph that is separated from the “rest of the graph” by a small-sized separator, and everything about H may be understood in terms of the graph induced by H itself and the limited interaction it has with $G \setminus H$ via its boundary vertices. If the size of the protrusion is large, we may want to replace it with another graph X that is much smaller but whose behaviour with respect to $G \setminus H$ is identical to H in the context of the problem that we are studying. Specifically, we would like that the solution to the problem in question does not change after we have made the replacement (or changes in a controlled manner that can be tracked as we make these replacements). This motivates us to define an equivalence that captures the essence of what we hope to do in terms the replacement. We would like to declare H equivalent to X if the size of the solution of G and $(G \setminus H) \cup^* X$ is exactly the same, where \cup^* is some notion of a replacement operation that we have not defined precisely yet. Notice, however, that a natural notion of replacement would leave the boundary vertices intact and perform a cut-and-paste on the rest of H . This is precisely what the protrusion based reduction rules do. Combined with some combinatorial properties of graphs this results in polynomial and in most cases linear kernels for variety of problems.

Overview of Meta Kernelization Results. Given a graph $G = (V, E)$, we define $\mathbf{B}_G^r(S)$ to be the set of all vertices of G whose distance from some vertex in S is at most r . Let \mathcal{G} be the family of planar graphs and let integer $k > 0$ be a parameter. We say that a parameterized problem $\Pi \subseteq \mathcal{G} \times \mathbb{N}$ is *compact* if there exist an integer r such that for all $(G = (V, E), k) \in \Pi$, there is a set $S \subseteq V$ such that $|S| \leq r \cdot k$, $\mathbf{B}_G^r(S) = V$ and $k \leq |V|^r$. Similarly, Π is *quasi-compact* if there exists an integer r such that for every $(G, k) \in \Pi$, there is a set $S \subseteq V$ such that $|S| \leq r \cdot k$, $\mathbf{tw}(G \setminus \mathbf{B}_G^r(S)) \leq r$ and $k \leq |V|^r$ where $\mathbf{tw}(G)$ denotes the treewidth of G . Notice that if a problem is compact then it is also quasi-compact. For ease of presentation the definitions of *compact* and *quasi-compact* are more restrictive here than in the following paper [7].

The following theorem from [7] yields linear kernels for a variety of problems on planar graphs. To this end they utilise the notion of *finite integer index*. This term first appeared in the work by Bodlaender and van Antwerpen-de Fluiter [14] and is similar to the notion of *finite state*. We first define the notion of t -boundary graphs and the gluing operation. A t -*boundary graph* is a graph $G = (V, E)$ with t distinguished vertices, uniquely labelled from 1 to t . The set $\partial(G)$ of labelled vertices is called the boundary of G . The vertices in $\partial(G)$ are referred to as boundary vertices or terminals. Let G_1 and G_2 be two t -boundary graphs. By $G_1 \oplus G_2$ we denote the t -boundary graph obtained by taking the disjoint union of G_1 and G_2 and identifying each vertex of $\partial(G_1)$ with the vertex of $\partial(G_2)$ with the same label; that is, we *glue* them together on the boundaries. In $G_1 \oplus G_2$ there is an edge between two labelled vertices if there is an edge

between them in G_1 or in G_2 . For a parameterized problem, Π on graphs in \mathcal{G} and two t -boundaried graphs G_1 and G_2 , we say that $G_1 \equiv_{\Pi} G_2$ if there exists a constant c such that for all t -boundaried graphs G_3 and for all k we have $G_1 \oplus G_3 \in \mathcal{G}$ if and only if $G_2 \oplus G_3 \in \mathcal{G}$ and $(G_1 \oplus G_3, k) \in \Pi$ if and only if $(G_2 \oplus G_3, k + c) \in \Pi$. Note that for every t , the relation \equiv_{Π} on t -boundaried graphs is an equivalence relation. A problem Π has *finite integer index* (FII), if and only if for every t , \equiv_{Π} is of finite index, that is, has a finite number of equivalence classes. Compact problems that have FII include DOMINATING SET and CONNECTED VERTEX COVER while FEEDBACK VERTEX SET has FII and is quasi-compact but not compact. We are now in position to state the theorem.

Theorem 5. *Let $\Pi \subseteq \mathcal{G} \times \mathbb{N}$ be quasi-compact and has FII. Then Π admits a linear kernel.*

Overview of the Methods. We give an outline of the main ideas used to prove Theorem 5. For a problem Π and an instance $(G = (V, E), k)$ the kernelization algorithm repeatedly identifies a part of the graph to reduce and replaces this part by smaller equivalent part. Since each such step decreases the number of vertices in the graph the process stops after at most $|V|$ iterations. In particular, the algorithm identifies a *constant size separator* S that cuts off a *large* chunk of the graph of *constant treewidth*. This chunk is then considered as a $|S|$ -boundaried graph $G' = (V', E')$ with boundary S . Let G^* be the other side of the separator, that is $G' \oplus G^* = G$. Since Π has FII there exists a finite set \mathcal{S} of $|S|$ -boundaried graphs such that $\mathcal{S} \subseteq \mathcal{G}$ and for any $|S|$ -boundaried graph G_1 there exists a $G_2 \in \mathcal{S}$ such that $G_2 \equiv_{\Pi} G_1$. The definition of “large chunk” is that G' should be larger than the largest graph in \mathcal{S} . Hence we can find a $|S|$ -boundaried graph $G_2 \in \mathcal{S}$ and a constant c such that $(G, k) = (G' \oplus G^*, k) \in \Pi$ if and only if $(G_2 \oplus G^*, k - c) \in \Pi$. The reduction is just to change (G, k) into $(G_2 \oplus G^*, k - c)$. Given G' we can identify G_2 in time linear in $|V'|$ by using the fact that G' has constant treewidth and that all graphs in \mathcal{S} have constant size.

We now proceed to analyze the size of any reduced yes-instance of Π . We show that if Π is compact (not quasi-compact), then the size of a reduced yes-instance (G, k) must be at most $O(k)$. Since $(G = (V, E), k) \in \Pi$ and Π is compact, there is an $O(k)$ sized set $S' \subseteq V$ such that $\mathbf{B}_G^r(S') = V$ for some constant r depending only on Π . One can show that if such a set S' exists there must exist another $O(k)$ sized set S such that the connected components of $G[V \setminus S]$ can be grouped into $O(k)$ chunks as described in the paragraph above. If any of these chunks have more vertices than the largest graph in \mathcal{S} we could have performed the reduction. This implies that any reduced yes-instance has size at most ck for some fixed constant c . Hence if a reduced instance is larger than ck the kernelization algorithm returns NO.

Finally to prove Theorem 5 even when Π is quasi-compact, they show that the set of reduced instances of a quasi-compact problem is in fact compact. Observe that it is the set of reduced instances that becomes compact and *not* Π itself. The main idea is that if $G = (V, E)$ has a set $S \subseteq V$ such that the treewidth of $G[V \setminus \mathbf{B}_G^r(S)]$ is constant and there exists a vertex v which is far away from S , then we can find a large subgraph to reduce.

The parameterized versions of many basic optimization problems have finite integer index, including problems like DOMINATING SET, (CONNECTED) r -DOMINATING SET, (CONNECTED) VERTEX COVER, FEEDBACK VERTEX SET, EDGE DOMINATING SET, INDEPENDENT SET, MIN LEAF SPANNING TREE, INDUCED MATCHING, TRIANGLE PACKING, CYCLE PACKING, MAXIMUM FULL-DEGREE SPANNING TREE, and many others [7,21].

There are problems like INDEPENDENT DOMINATING SET, LONGEST PATH, LONGEST CYCLE, MAXIMUM CUT, MINIMUM COVERING BY CLIQUES, INDEPENDENT DOMINATING SET, and MINIMUM LEAF OUT-BRANCHING and various edge packing problems which are known not to have FII [21]. It was shown in [7] that compact problems expressible in an extension of Monadic Second Order Logic, namely Counting Monadic Second Order Logic, have polynomial kernels on planar graphs. This implies polynomial kernels for INDEPENDENT DOMINATING SET, MINIMUM LEAF OUT-BRANCHING, and some edge packing problems on planar graphs. The results from [7] hold not only for planar graphs but for graphs of bounded genus. It was shown in [37] that if instead of quasi-compactness, we request another combinatorial property, bidimensionality with certain separability properties, then an analogue of Theorem 5 can be obtained for much more general graph classes, like graphs excluding some fixed (apex) graph as a minor.

Bodlaender et al. [7] were the first to use protrusion techniques (or rather graph reduction techniques) to obtain kernels, but the idea of using graph replacement for algorithms has been around for long time. The idea of graph replacement for algorithms dates back to Fellows and Langston [31]. Arnborg et al. [6] essentially showed that effective protrusion reduction procedures exist for many problems on graphs of bounded treewidth. Using this, Arnborg et al. [6] obtained a linear time algorithm for MSO expressible problems on graphs of bounded treewidth. Bodlaender and Fluiter [8,14,21] generalized these ideas in several ways — in particular, they applied it to some optimization problems. It is also important to mention the work of Bodlaender and Hagerup [10], who used the concept of graph reduction to obtain parallel algorithms for MSO expressible problems on bounded treewidth graphs.

4.2 Algebraic and Probabilistic Methods

A r -CNF formula $F = c_1 \wedge \cdots \wedge c_m$ on variable set $V(F)$ is a boolean formula where each clause has size exactly r and each clause is a disjunction of literals. In the parameterized MAX- r -SAT problem

p -MAX- r -SAT

Instance: A r -CNF formula F , and a non-negative integer k .

Parameter: k

Problem: Decide whether F has a truth assignment satisfying at least k clauses.

Observe that the expected number of clauses satisfied by a random truth assignment that sets each variable of F to one or zero is equal to

$$\mu_F = (1 - 2^{-r})m$$

and thus there is always an assignment satisfying at least μ_F clauses. This implies that at least $m/2$ clauses are always satisfied and hence this parameterization of MAX- r -SAT always has a polynomial kernel because of the following argument. If $k \leq m/2$ then the answer is yes else we have that $m \leq 2k$ and hence $n \leq 2kr$. Thus given a r -CNF formula F , the more meaningful question is whether there exists a truth assignment for F satisfying at least $\mu_F + k$ clauses. We call this version of the MAX- r -SAT problem as p -AG-MAX- r -SAT, that is, problem where the parameterization is beyond the guaranteed lower bound on the solution.

p -AG-MAX- r -SAT

Instance: A r -CNF formula F , and a non-negative integer k .

Parameter: k

Problem: Decide whether F has a truth assignment satisfying at least $\mu_F + k$ clauses.

The parameterized study of problems above a guaranteed lower bound was initiated by Mahajan and Raman [54]. They showed that several above guarantee versions of MAX-CUT and MAX-SAT are FPT and provided a number of open problems around parameterizations beyond guaranteed lower and upper bounds. In a breakthrough paper Gutin et al [42] developed a probabilistic approach to problems parameterized above or below tight bounds. Alon et al. [3] combined this approach with methods from algebraic combinatorics and Fourier analysis to obtain FPT algorithm for parameterized MAX- r -SAT beyond the guaranteed lower bound. Other significant results in this direction include quadratic kernels for ternary permutation constraint satisfaction problems parameterized above average and results around system of linear equations modulo 2 [19,43]. In what follows we outline the method and then illustrate the method using an example.

Informal Description of the Method. We give a brief description of the probabilistic method with respect to a given problem Π parameterized above a tight lower bound or below a tight upper bound. We first apply some reductions rules to reduce Π to its special case Π' . Then we introduce a random variable X such that the answer to Π is YES if and only if X takes, with positive probability, a value greater or equal to the parameter k . Now using some probabilistic inequalities on X , we derive upper bounds on the size of NO-instances of Π' in terms of a function of the parameter k . If the size of a given instance exceeds this bound, then we know the answer is YES; otherwise, we produce a problem kernel.

Probabilistic Inequalities. A random variable is *discrete* if its distribution function has a finite or countable number of positive increases. A random variable X

is a *symmetric* if $-X$ has the same distribution function as X . If X is discrete, then X is symmetric if and only if $\text{Prob}(X = a) = \text{Prob}(X = -a)$ for each real a . Let X be a symmetric variable for which the first moment $\mathbb{E}(X)$ exists. Then $\mathbb{E}(X) = \mathbb{E}(-X) = -\mathbb{E}(X)$ and, thus, $\mathbb{E}(X) = 0$. The following is easy to prove [42].

Lemma 6. *If X is a symmetric random variable and $\mathbb{E}(X^2) < \infty$, then*

$$\text{Prob}(X \geq \sqrt{\mathbb{E}(X^2)}) > 0.$$

Unfortunately, often X is not symmetric, but Lemma 7 provides an inequality that can be used in many such cases. This lemma was proved by Alon et al. [4]; a weaker version was obtained by Håstad and Venkatesh [45].

Lemma 7. *Let X be a random variable and suppose that its first, second and fourth moments satisfy $\mathbb{E}(X) = 0$, $\mathbb{E}(X^2) = \sigma^2 > 0$ and $\mathbb{E}(X^4) \leq b\sigma^4$, respectively. Then $\text{Prob}(X > \frac{\sigma}{4\sqrt{b}}) \geq \frac{1}{4^{4/3}b}$.*

Since it is often rather nontrivial to evaluate $\mathbb{E}(X^4)$ in order to check whether $\mathbb{E}(X^4) \leq b\sigma^4$ holds, one can sometimes use the following extension of Khinchin's Inequality by Bourgain [15].

Lemma 8. *Let $f = f(x_1, \dots, x_n)$ be a polynomial of degree r in n variables x_1, \dots, x_n with domain $\{-1, 1\}$. Define a random variable X by choosing a vector $(\epsilon_1, \dots, \epsilon_n) \in \{-1, 1\}^n$ uniformly at random and setting $X = f(\epsilon_1, \dots, \epsilon_n)$. Then, for every $p \geq 2$, there is a constant c_p such that*

$$(\mathbb{E}(|X|^p))^{1/p} \leq (c_p)^r (\mathbb{E}(X^2))^{1/2}.$$

In particular, $c_4 \leq 2^{3/2}$.

An Illustration. Consider the following problem: given a digraph $D = (V, A)$ and a positive integer k , does there exist an acyclic subdigraph of D with at least k arcs? It is easy to prove that this parameterized problem has a linear kernel. Observe that D always has an acyclic subdigraph with at least $|A|/2$ arcs. Indeed, consider a bijection $\alpha : V \rightarrow \{1, \dots, |V|\}$ and the following subdigraphs of D : $(V, \{xy \in A : \alpha(x) < \alpha(y)\})$ and $(V, \{xy \in A : \alpha(x) > \alpha(y)\})$. Both subdigraphs are acyclic and at least one of them has at least $|A|/2$ arcs. Thus the input D itself is a kernel with $2k$ arcs and at most $4k$ vertices. Thus a more natural interesting parameterization is following: decide whether $D = (V, A)$ contains an acyclic subdigraph with at least $|A|/2 + k$ arcs. We choose $|A|/2 + k$ because $|A|/2$ is a *tight lower bound* on the size of a largest acyclic subdigraph. Indeed, the size of a largest acyclic subdigraph of a symmetric digraph $D = (V, A)$ is precisely $|A|/2$. A digraph $D = (V, A)$ is *symmetric* if $xy \in A$ implies $yx \in A$. More precisely we study the following problem.

p -LINEAR ORDERING ABOVE TIGHT LOWER BOUND (LOALB)

Instance: A digraph D with each arc ij with integer positive weight w_{ij} , and a positive integer k .

Parameter: k

Problem: Decide whether there is an acyclic subdigraph of D of weight at least $W/2 + k$, where $W = \sum_{ij \in A} w_{ij}$.

Consider the following reduction rule:

Reduction Rule 1. Assume D has a directed 2-cycle iji ; if $w_{ij} = w_{ji}$ delete the cycle, if $w_{ij} > w_{ji}$ delete the arc ji and replace w_{ij} by $w_{ij} - w_{ji}$, and if $w_{ji} > w_{ij}$ delete the arc ij and replace w_{ji} by $w_{ji} - w_{ij}$.

It is easy to check that the answer to LOALB for a digraph D is YES if and only if the answer to LOALB is YES for a digraph obtained from D using the reduction rule as long as possible.

Let $D = (V, A)$ be an oriented graph, let $n = |V|$ and $W = \sum_{ij \in A} w_{ij}$. Consider a random bijection: $\alpha : V \rightarrow \{1, \dots, n\}$ and a random variable $X(\alpha) = \frac{1}{2} \sum_{ij \in A} \epsilon_{ij}(\alpha)$, where $\epsilon_{ij}(\alpha) = w_{ij}$ if $\alpha(i) < \alpha(j)$ and $\epsilon_{ij}(\alpha) = -w_{ij}$, otherwise. It is easy to see that $X(\alpha) = \sum \{w_{ij} : ij \in A, \alpha(i) < \alpha(j)\} - W/2$. Thus, the answer to LOALB is YES if and only if there is a bijection $\alpha : V \rightarrow \{1, \dots, n\}$ such that $X(\alpha) \geq k$. Since $\mathbb{E}(\epsilon_{ij}) = 0$, we have $\mathbb{E}(X) = 0$. Let $W^{(2)} = \sum_{ij \in A} w_{ij}^2$. Then one can prove the following:

Lemma 9 ([42]). $\mathbb{E}(X^2) \geq W^{(2)}/12$.

Using Lemma 9 we prove the following main result of this section.

Theorem 6 ([42]). *The problem LOALB admits a kernel with $O(k^2)$ arcs.*

Proof. Let H be a digraph. We know that the answer to LOALB for H is YES if and only if the answer to LOALB is YES for a digraph D obtained from H using Reduction Rule 1 as long as possible. Observe that D is an oriented graph. Let \mathcal{B} be the set of bijections from V to $\{1, \dots, n\}$. Observe that $f : \mathcal{B} \rightarrow \mathcal{B}$ such that $f(\alpha(v)) = |V| + 1 - \alpha(v)$ for each $\alpha \in \mathcal{B}$ is a bijection. Note that $X(f(\alpha)) = -X(\alpha)$ for each $\alpha \in \mathcal{B}$. Therefore, $\text{Prob}(X = a) = \text{Prob}(X = -a)$ for each real a and, thus, X is symmetric. Thus, by Lemmas 6 and 9, we have $\text{Prob}(X \geq \sqrt{W^{(2)}/12}) > 0$. Hence, if $\sqrt{W^{(2)}/12} \geq k$, there is a bijection $\alpha : V \rightarrow \{1, \dots, n\}$ such that $X(\alpha) \geq k$ and, thus, the answer to LOALB (for both D and H) is YES. Otherwise, $|A| \leq W^{(2)} < 12 \cdot k^2$. \square

4.3 Randomized Kernels

A question whether the following problem has a polynomial kernel or not had remained elusive for a few years until recently.

p -ODD CYCLE TRANSVERSAL

Instance: An undirected graph $G = (V, E)$ and a positive integer k .

Parameter: k

Problem: Decide whether there exist a set $S \subseteq V$ such that $G \setminus S$ does not contain odd cycles?

Using techniques from matroid theory it has been recently shown that this problem admits a randomized polynomial kernel [52]. The main part of this kernelization algorithm is to adapt the steps in the FPT algorithm for ODD CYCLE TRANSVERSAL as “independent sets” of the matroid called “gammoid”. This exploits the duality between max-flow and min-cut. Recently using another technique from matroid theory a combinatorial kernel has been proposed [51]. This approach works for several other problems including ALMOST-2-SAT. Even a short description on this algorithm is beyond the scope of this article. We refer the interested readers to the following articles [52,51].

5 Lower Bound Machinery

Lemma 1 implies that a problem has a kernel if and only if it is fixed parameter tractable. However, we are interested in kernels that are as small as possible, and a kernel obtained using Lemma 1 has size that equals the dependence on k in the running time of the best known FPT algorithm for the problem. The question is — can we do better? In particular, can we get polynomial sized kernels for problems that admit FPT algorithms? The answer is that quite often we can, as we saw in the previous section, but it turns out that there are a number of problems which are unlikely to have polynomial kernels. It is only very recently that a methodology to rule out polynomial kernels has been developed [9,38]. The existence of polynomial kernels are ruled out, in this framework, by linking the availability of a polynomial kernel to an unlikely collapse in classical complexity. These developments deepen the connection between classical and parameterized complexity.

In this section we survey the techniques that have been developed to show kernelization lower bounds. To begin with, we consider the following problem.

p-LONGEST PATH

Instance: An undirected graph $G = (V, E)$ and a non-negative integer k .

Parameter: k

Problem: Does G have a path of length k ?

It is well known that the *p*-LONGEST PATH problem can be solved in time $O(c^k n^{O(1)})$ using the well known method of COLOR-CODING [5]. Is it feasible that it also admits a polynomial kernel? We argue that intuitively this should not be possible. Consider a large set $(G_1, k), (G_2, k), \dots, (G_t, k)$ of instances to the *p*-LONGEST PATH problem. If we make a new graph G by just taking the disjoint union of the graphs G_1, \dots, G_t we see that G contains a path of length k if and only if G_i contains a path of length k for some $i \leq t$. Suppose the *p*-LONGEST PATH problem had a polynomial kernel, and we ran the kernelization algorithm on G . Then this algorithm would in polynomial time return a new instance $(G' = (V', E'), k')$ such that $|V'| = k^{O(1)}$, a number potentially much smaller than t . This means that in some sense, the kernelization algorithm considers the instances $(G_1, k), (G_2, k), \dots, (G_t, k)$ and in *polynomial time* figures out which of the instances are the most likely to contain a path of length k . However, at least

intuitively, this seems almost as difficult as solving the instances themselves and since the p -LONGEST PATH problem is NP-complete, this seems unlikely. We now formalize this intuition.

Definition 3. [Distillation [9]]

- An *OR-distillation algorithm* for a language $L \subseteq \Sigma^*$ is an algorithm that receives as input a sequence x_1, \dots, x_t , with $x_i \in \Sigma^*$ for each $1 \leq i \leq t$, uses time polynomial in $\sum_{i=1}^t |x_i|$, and outputs $y \in \Sigma^*$ with (a) $y \in L \iff x_i \in L$ for some $1 \leq i \leq t$ and (b) $|y|$ is polynomial in $\max_{i \leq t} |x_i|$. A language L is *OR-distillable* if there is a *OR-distillation algorithm* for it.
- An *AND-distillation algorithm* for a language $L \subseteq \Sigma^*$ is an algorithm that receives as input a sequence x_1, \dots, x_t , with $x_i \in \Sigma^*$ for each $1 \leq i \leq t$, uses time polynomial in $\sum_{i=1}^t |x_i|$, and outputs $y \in \Sigma^*$ with (a) $y \in L \iff x_i \in L$ for all $1 \leq i \leq t$ and (b) $|y|$ is polynomial in $\max_{i \leq t} |x_i|$. A language L is *AND-distillable* if there is an *AND-distillation algorithm* for it.

Observe that the notion of distillation is defined for unparameterized problems. Bodlaender et al. [9] conjectured that no NP-complete language can have an OR-distillation or an AND-distillation algorithm.

Conjecture 1. [OR-Distillation Conjecture [9]] No NP-complete language L is OR-distillable.

Conjecture 2. [AND-Distillation Conjecture [9]] No NP-complete language L is AND-distillable.

One should notice that if any NP-complete language is distillable, then so are all of them. Fortnow and Santhanam [38] were able to connect the OR-Distillation Conjecture to a well-known conjecture in classical complexity. In particular they proved that if the OR-Distillation Conjecture fails, then $coNP \subseteq NP/poly$, implying that the *polynomial time hierarchy* [59] collapses to the third level, a collapse that is deemed unlikely. Until very recently, establishing a similar connection for the AND-Distillation Conjecture was one of the central open problems of the area. It is now established that both conjectures hold up to reasonable complexity-theoretic assumptions (see also Section 6.4).

Theorem 7 ([38,28])

- If the *OR-Distillation Conjecture* fails, then $coNP \subseteq NP/poly$.
- If the *AND-Distillation Conjecture* fails, then $coNP \subseteq NP/poly$.

We are now ready to define the parameterized analogue of distillation algorithms and connect this notion to the Conjectures 1 and 2.

Definition 4. [Composition [9]]

- A *composition algorithm* (also called *OR-composition algorithm*) for a parameterized problem $\Pi \subseteq \Sigma^* \times \mathbb{N}$ is an algorithm that receives as input a

sequence $((x_1, k), \dots, (x_t, k))$, with $(x_i, k) \in \Sigma^* \times \mathbb{N}^+$ for each $1 \leq i \leq t$, uses time polynomial in $\sum_{i=1}^t |x_i| + k$, and outputs $(y, k') \in \Sigma^* \times \mathbb{N}^+$ with (a) $(y, k') \in \Pi \iff (x_i, k) \in \Pi$ for some $1 \leq i \leq t$ and (b) k' is polynomial in k . A parameterized problem is compositional (or OR-compositional) if there is a composition algorithm for it.

- An AND-composition algorithm for a parameterized problem $\Pi \subseteq \Sigma^* \times \mathbb{N}$ is an algorithm that receives as input a sequence $((x_1, k), \dots, (x_t, k))$, with $(x_i, k) \in \Sigma^* \times \mathbb{N}^+$ for each $1 \leq i \leq t$, uses time polynomial in $\sum_{i=1}^t |x_i| + k$, and outputs $(y, k') \in \Sigma^* \times \mathbb{N}^+$ with (a) $(y, k') \in \Pi \iff (x_i, k) \in \Pi$ for all $1 \leq i \leq t$ and (b) k' is polynomial in k . A parameterized problem is AND-compositional if there is an AND-composition algorithm for it.

Composition and distillation algorithms are very similar. The main difference between the two notions is that the restriction on output size for distillation algorithms is replaced by a restriction on the parameter size for the instance the composition algorithm outputs. We define the notion of the *unparameterized version* of a parameterized problem L . The mapping of parameterized problems to unparameterized problems is done by mapping (x, k) to the string $x\#1^k$, where $\# \notin \Sigma$ denotes the blank letter and 1 is an arbitrary letter in Σ . In this way, the unparameterized version of a parameterized problem Π is the language $\tilde{\Pi} = \{x\#1^k \mid (x, k) \in \Pi\}$. The following theorem yields the desired connection between the two notions.

Theorem 8 ([9,28]). *Let Π be a compositional parameterized problem whose unparameterized version $\tilde{\Pi}$ is NP-complete. Then, if Π has a polynomial kernel then $coNP \subseteq NP/poly$. Similarly, let Π be an AND-compositional parameterized problem whose unparameterized version $\tilde{\Pi}$ is NP-complete. Then, if Π has a polynomial kernel, $coNP \subseteq NP/poly$.*

We can now formalize the discussion from the beginning of this section.

Theorem 9 ([9]). *p -LONGEST PATH does not admit a polynomial kernel unless $coNP \subseteq NP/poly$.*

Proof. The unparameterized version of p -LONGEST PATH is known to be NP-complete [39]. We now give a composition algorithm for the problem. Given a sequence $(G_1, k), \dots, (G_t, k)$ of instances we output (G, k) where G is the disjoint union of G_1, \dots, G_t . Clearly G contains a path of length k if and only if G_i contains a path of length k for some $i \leq t$. By Theorem 8 p -LONGEST PATH does not have a polynomial kernel unless $coNP \subseteq NP/poly$. □

An identical proof can be used to show that the p -LONGEST CYCLE problem does not admit a polynomial kernel unless $coNP \subseteq NP/poly$. For many problems, it is easy to give AND-composition algorithms. For instance, the “disjoint union” trick yields AND-composition algorithms for the p -TREEWIDTH, p -PATHWIDTH and p -CUTWIDTH problems, among many others. Coupled with Theorem 8 this implies that these problems do not admit polynomial kernels unless $coNP \subseteq NP/poly$.

For some problems, obtaining a composition algorithm directly is a difficult task. Instead, we can give a reduction from a problem that provably has no polynomial kernel unless $coNP \subseteq NP/poly$ to the problem in question such that a polynomial kernel for the problem considered would give a kernel for the problem we reduced from. We now define the notion of *polynomial parameter transformations*.

Definition 5 ([13]). *Let P and Q be parameterized problems. We say that P is polynomial parameter reducible to Q , written $P \leq_{ppt} Q$, if there exists a polynomial time computable function $f : \Sigma^* \times \mathbb{N} \rightarrow \Sigma^* \times \mathbb{N}$ and a polynomial p , such that for all $(x, k) \in \Sigma^* \times \mathbb{N}$ (a) $(x, k) \in P$ if and only if $(x', k') = f(x, k) \in Q$ and (b) $k' \leq p(k)$. The function f is called polynomial parameter transformation.*

Proposition 1 ([13]). *Let P and Q be the parameterized problems and \tilde{P} and \tilde{Q} be the unparameterized versions of P and Q respectively. Suppose that \tilde{P} is NP-complete and \tilde{Q} is in NP. Furthermore if there is a polynomial parameter transformation from P to Q , then if Q has a polynomial kernel then P also has a polynomial kernel.*

Proposition 1 shows how to use polynomial parameter transformations to show kernelization lower bounds. A notion similar to polynomial parameter transformation was independently used by Fernau et al. [33] albeit without being explicitly defined. We now give an example of how Proposition 1 can be useful for showing that a problem does not admit a polynomial kernel. In particular, we show that the p -PATH PACKING problem does not admit a polynomial kernel unless $coNP \subseteq NP/poly$. In this problem you are given a graph G together with an integer k and asked whether there exists a collection of k mutually vertex-disjoint paths of length k in G . This problem is known to be fixed parameter tractable [5] and is easy to see that for this problem the “disjoint union” trick discussed earlier does not directly apply. Thus we resort to polynomial parameter transformations.

Theorem 10. *p -PATH PACKING does not admit a polynomial kernel unless $coNP \subseteq NP/poly$.*

Proof. We give a polynomial parameter transformation from the p -LONGEST PATH problem. Given an instance (G, k) to p -LONGEST PATH we construct a graph G' from G by adding $k - 1$ vertex disjoint paths of length k . Now G contains a path of length k if and only if G' contains k paths of length k . This concludes the proof. \square

6 Recent Developments in Lower Bounds

In this section, we provide a brief exposition of some of the more recent developments that have emerged in pursuing lower bounds, namely, cross-compositions, the notion of co-nondeterminism in compositions, and the development that linked the failure of the AND conjecture with an unexpected collapse in classical complexity.

6.1 Cross Composition

Recall that an OR-composition algorithm works by composing multiple instances of a parameterized problem Q into a single instance of \tilde{Q} with a parameter value bounded by a polynomial function of k , the common parameter of all input instances. Further, we also had the constraint that the parameter of the output instance may not depend on the size of the largest input instance, and also should be independent of the number of instances that are input to the algorithm.

It turns out that a variation of the OR-composition algorithm, where the requirements on the output instance are more “relaxed”, can still be used to argue lower bounds. This variant was introduced in [11], and is called *cross-composition*. The technique is akin to OR-composition to the extent that it is meant to output the boolean OR of a number of instances. On the other hand, a cross-composition is less restrictive than the standard OR-composition in various ways:

- The source and target problem of the composition need no longer be the same.
- The input to a cross-composition algorithm is a list of classical instances instead of parameterized instances, the inputs do not have a parameter in which the output parameter of the composition must be bounded; instead we require that the size of the output parameter is polynomially bounded in the size of the largest input instance.
- The output parameter may depend polynomially on the logarithm of the number of input instances.

With cross-composition, it is sufficient to compose (via a boolean OR) any classical NP-hard problem into an instance of the parameterized problem Q for which we want to prove a lower-bound, and the parameter of the output instance is permitted to depend on the number of input instances, and the size of the largest instance as well.

For establishing the technique of cross-composition, the notion of a *polynomial equivalence relation* is introduced. Informally, an equivalence relation on Σ^* is a polynomial equivalence relation if it can be “identified” in polynomial time and if the number of equivalence classes of any finite subset are polynomially many in the maximum element of the subset. The formal definition is the following:

Definition 6 (Polynomial equivalence relation, [11]). *An equivalence relation \mathcal{R} on Σ^* is called a polynomial equivalence relation if the following two conditions hold:*

1. *There is an algorithm that given two strings $x, y \in \Sigma^*$ decides whether x and y belong to the same equivalence class in $(|x| + |y|)^{O(1)}$ time.*
2. *For any finite set $S \subseteq \Sigma^*$ the equivalence relation \mathcal{R} partitions the elements of S into at most $(\max_{x \in S} |x|)^{O(1)}$ classes.*

We now turn to the definition of cross-composition:

Definition 7 ([11]). Let $L \subseteq \Sigma^*$ be a set and let $Q \subseteq \Sigma^* \times N$ be a parameterized problem. We say that L cross-composes into Q if there is a polynomial equivalence relation \mathcal{R} and an algorithm which, given t strings x_1, x_2, \dots, x_t belonging to the same equivalence class of \mathcal{R} , computes an instance $(x, k) \in \Sigma^* \times N$ in time polynomial in $\sum_{i=1}^t |x_i|$ such that:

1. $(x, k) \in Q \Rightarrow x_i \in L$ for some $1 \leq i \leq t$,
2. k is bounded by a polynomial in $\max_{i=1}^t |x_i| + \log t$.

The existence of a cross-composition from a NP-complete problem into a parameterized problem implies kernel lower bounds for the parameterized problem because a distillation for SAT can be inferred from the cross-composition and the assumption of a polynomial kernel for the parameterized problem. Recall that the existence of a distillation for any NP-complete problem implies that $coNP \subseteq NP/poly$, which completes the argument for the infeasibility of polynomial kernels for problems that admit a cross-composition. Formally, we would say that unless $coNP \subseteq NP/poly$, a problem that admits a cross-composition does not have apolynomial kernel. We now turn to an overview of the argument that leads to a distillation starting from a cross-composition and a polynomial kernel.

Assume that we have a cross-composition from a NP-complete language L to a parameterized language Q . Let m denote the size of the largest input to the distillation algorithm. We describe informally how a cross-composition and a polynomial kernel for Q can be used to devise a distillation algorithm for SAT. For a more formal argument, the reader is referred to [11].

- First, duplicate instances are eliminated from the sequence of inputs to ensure that $t \leq (|\Sigma| + 1)^m$, or that $\log t \in O(m)$. All instances of SAT are transformed into equivalent instances of L (this can be done since L is NP-complete) — note that the sizes of the instances of L are also polynomial in m .
- We now pairwise compare instances using the polynomial-time equivalence test of \mathcal{R} (whose existence is guaranteed by the definition of a cross-composition) to partition the L -instances (y_1, \dots, y_t) into partite sets Y_1, \dots, Y_r such that all instances from the same partite set are equivalent under \mathcal{R} . The properties of a polynomial equivalence relation guarantee that r is polynomial in m and that this partitioning step takes polynomial time in the total input size.
- Subsequently, a cross-composition is applied to each group of instances in Y_i . In all the parameterized instances that are output by the cross-composition, we have that the parameter is a polynomial function of m , since $\log t \in O(m)$.
- We now apply the kernelization algorithm to obtain polynomial kernels for each instance of Q that is output by the cross-composition. Note that there are polynomially many instances, and each instance after kernelization is also polynomial in size.
- These instances can now be converted back to SAT instances, which can be combined in a straightforward manner to a single instance reflecting the Boolean OR of the original sequence of instances.

Having established what a cross-composition algorithm is, and why it implies kernel lower bounds, we now state some applications of this technique. In [11], the problems considered include p -CHROMATIC NUMBER and p -CLIQUE parameterized by vertex cover number and p -FEEDBACK VERTEX SET parameterized by deletion distance to cluster graphs or co-cluster graphs.

In the case of p -CLIQUE it was already known [9] that the problem does not admit a polynomial kernel parameterized by the treewidth of the graph; since the vertex cover number is at least as large as the treewidth, this is a stronger result. For the unweighted p -FEEDBACK VERTEX SET problem, which admits a polynomial kernel parameterized by the target size of the feedback set [16,60], it can be shown, using cross-composition, that there is no polynomial kernel for the parameterization by deletion distance to cluster graphs or co-cluster graphs.

6.2 Finer Lower Bounds

In [23], the kernel lower bound established by Theorem 8 was generalized further to provide for lower bounds based on different polynomial functions for the kernel size. The OR of a language L is the language $\text{OR}(L)$ that consists of all tuples (x_1, \dots, x_t) for which there is an $i \in \{1, \dots, t\}$ with $x_i \in L$. Instance $x = (x_1, \dots, x_t)$ for $\text{OR}(L)$ has two parameters: the length t of the tuple and the maximum bitlength $s = \max_i |x_i|$ of the individual instance for L . The following lemma was established in [23] to prove conditional lower bounds on the kernel sizes.

Lemma 10 ([23]). *Let Π be a problem parameterized by k and let L be an NP-hard problem. Assume that there is a polynomial-time mapping reduction f from $\text{OR}(L)$ to Π and a number $d > 0$ with the following property: given an instance $x = (x_1, \dots, x_t)$ for $\text{OR}(L)$ in which each x_i has size at most s , the reduction produces an instance $f(x)$ for whose parameter k is at most $t^{\frac{1}{d}+o(1)} \cdot \text{poly}(s)$. Then L does not have kernels of size $O(k^{d-\varepsilon})$ for any $\varepsilon > 0$ unless $\text{coNP} \subseteq \text{NP}/\text{poly}$.*

Bodlaender et al. [9] formulated this method without the dependency on t . This suffices to prove polynomial kernel lower bounds since d can be chosen as an arbitrarily large constant. It was observed in [23] that the proofs in [9,38] can be easily adapted to obtain the formulation above, and that it can be generalized to an oracle communication setting. See [23,22] for more details.

We describe an application of the lemma above to the problem of p -VERTEX COVER in graphs, where we have $d = 2$. Following the presentation in [22], we set L to be the p -MULTICOLORED BICLIQUE problem, where the input is a bipartite graph B on the vertex set $U \cup W$, an integer k , and partitions of U and W into k parts, namely (U_1, \dots, U_k) and (W_1, \dots, W_k) , respectively. We wish to decide if B contains a biclique $K_{k,k}$ that has one vertex from each U_i and W_i for $1 \leq i \leq k$. This is a problem on bipartite graphs and it is NP-complete [22].

Theorem 11 ([23,22]). *p -VERTEX COVER does not have kernels of size $O(k^{2-\varepsilon})$ unless $\text{coNP} \subseteq \text{NP}/\text{poly}$.*

Proof. We apply Lemma 10 where we set L to be p -MULTICOLORED BICLIQUE. Given an instance (B_1, \dots, B_t) for $OR(L)$, we can assume that every instance B_i has the same number k of groups in the partitions and every group in every instance B_i has the same size n : by simple padding arguments. Furthermore, we can assume that \sqrt{t} is an integer. In the following, we refer to the t instances of p -MULTICOLORED BICLIQUE in the $OR(L)$ instance as $B_{(i,j)}$ for $1 \leq i; j \leq \sqrt{t}$; let $U_{(i,j)}$ and $W_{(i,j)}$ be the two bipartite classes of $B_{(i,j)}$.

First, we modify each instance $B_{(i,j)}$ in such a way that $U_{(i,j)}$ and $W_{(i,j)}$ become complete k -partite graphs: if two vertices $U_{(i,j)}$ or two vertices in $W_{(i,j)}$ are in different groups, then we make them adjacent. It is clear that there is a $2k$ -clique in the new graph $B'_{(i,j)}$ if and only if there is a correctly partitioned $K_{k,k}$ in $B_{(i,j)}$. We construct a graph G by introducing $2\sqrt{t}$ sets $(U^1, \dots, U^{\sqrt{t}})$, $W^1, \dots, W^{\sqrt{t}}$ of kn vertices each. For every $1 \leq i \leq j \leq \sqrt{t}$, we copy the graph $B'_{(i,j)}$ to the vertex set $U^i \cup W^j$ by mapping $U_{(i,j)}$ to U^i and $W_{(i,j)}$ to W^j . Note that $U_{(i,j)}$ and $W_{(i,j)}$ induces the same complete k -partite graph in $B'_{(i,j)}$ for every i and j , thus this copying can be done in such a way that $G[U^i]$ receives the same set of edges when copying $B'_{(i,j)}$ for any j (and similarly for $G[W^j]$). Therefore, $G[U^i \cup W^j]$ is isomorphic to $B'_{(i,j)}$ for every $1 \leq i \leq j \leq \sqrt{t}$.

It can be verified that G has a $2k$ -clique if and only if at least one $B'_{(i,j)}$ has a $2k$ -clique (and therefore at least one $B_{(i,j)}$ has a correctly partitioned $K_{k,k}$).

Let $N = 2\sqrt{t}kn$ be the number of vertices in G . Note that $N = t^{1/2} \cdot poly(s)$, where s is the maximum bitlength of the t instances in the $OR(L)$ instance. The graph G has a $2k$ -clique if and only if its complement \bar{G} has a vertex cover of size $N - 2k$. Thus $OR(L)$ can be reduced to an instance of p -VERTEX COVER with parameter at most $t^{1/2} \cdot poly(s)$, as required. \square

6.3 Co-nondeterminism in Compositions

In [50], the notion of co-nondeterministic composition is introduced, and it is shown that this concept excludes polynomial kernels, assuming $coNP \not\subseteq NP/poly$. The technique was applied to show that the $RAMSEY(k)$ problem does not admit a polynomial kernel. This is an interesting question posed by Rod Downey — and it asks if the following combination of the well-known CLIQUE and INDEPENDENT SET, known to be NP-complete and FPT, admits a polynomial kernel.

RAMSEY(k)

Instance: An undirected graph G and a non-negative integer k .

Parameter: k .

Problem: Does G contain an independent set or a clique of size k ?

Unlike for p -LONGEST PATH [9] (see also Section 5), the disjoint union of t instances of $RAMSEY(k)$ does not work satisfactorily as a composition algorithm (and neither would a join of the instances) as it would contain independent sets

of size $\omega(t)$. The intricate Packing Lemma due to Dell and van Melkebeek [23, Lemma 1], although designed in a different context, does not seem to be applicable either as it constructs an n -partite graph containing independent sets of size n which cannot be bounded in $O(\log t)$ when $t := t(n)$ is polynomially-bounded. Generally, it appears to be unlikely that one could pack the instances in such a way that solutions are confined to a part representing a single original instance.

In the context of establishing lower bounds for this problem, the notion of co-nondeterminism in compositions was formulated. A “co-nondeterministic” composition is formally defined as follows:

Definition 8. *Let $Q \subseteq \Sigma^* \times N$. A co-nondeterministic polynomial-time algorithm C is a coNP composition for Q if there is a polynomial p such that on input of t instances $(x_1, k), \dots, (x_t, k) \in \Sigma^* \times N$ the algorithm C takes time polynomial in $\sum_{i=1}^t |x_i|$ and outputs on each computation path an instance $(y, k') \subseteq \Sigma^* \times N$ with $k' \leq t^{o(1)}p(k)$ and such that the following holds:*

- *If at least one instance (x_i, k) is a yes-instance then all computation paths lead to the output of a yes-instance (y, k') .*
- *Otherwise, if all instances (x_i, k) are no-instances, then at least one computation path leads to the output of a no-instance.*

The main tool for establishing that the existence of a coNP composition for a parameterized problem implies a polynomial kernel lower bound for it is a lemma due to Dell and van Melkebeek [23].

It turns out that the combination of a co-nondeterministic composition and a polynomial kernel for a parameterized problem (whose classical version is NP-complete) implies the existence of an oracle communication protocol of a suitable kind. This further implies that the corresponding classical problem is in $coNP/poly$, and that finally leads us to the conclusion that $NP \subseteq coNP/poly$, establishing the lower bound.

6.4 The AND Conjecture

As we explained in Section 5, the failing of the AND conjecture did not have any significant implications in classical complexity. If it did have a connection analogous to the one that is enjoyed by the OR conjecture, then this would have several implications in settling the status of the kernelization complexity of a number of problems, to the extent that we make assumptions that are reasonable in the context of classical complexity. For example, in [20], it is shown that p -EDGE CLIQUE COVER has no polynomial kernel unless the AND conjecture fails. A number of AND-based-compositions exist for graph layout problems like p -TREEWIDTH, p -PATHWIDTH, p -CUTWIDTH and other problems like p -INDEPENDENT SET, p -DOMINATING SET when parameterized by the treewidth of the input graph. A more comprehensive discussion can be found in [9]. With this new development, all these problems do not have polynomial kernels unless $NP \subseteq coNP/poly$.

In a talk titled *On the Hardness of Compressing an AND of SAT Instances*, Andrew Drucker revealed that efficient AND-compression would also imply that $NP \subseteq coNP/poly$. To prove this result (and some extensions), any compression scheme is interpreted as a communication channel so as to exploit a certain bottleneck. This entails a new method to “disguise” information being fed into a compressive mapping. At the time of this writing, this work is unpublished, but the details that are available can be found in [28].

7 Conclusion and Discussion

In this section we mention several directions of possible development of kernelization.

Parameterization Vs Parameterizations. In parameterized complexity there are many reasonable possibilities to “parameterize a problem”. For an example for a graph optimization problem a parameter could be the solution size, the structure of the graph (like treewidth or pathwidth), distance of the graph from some polynomially solvable subclasses (for an example deleting at most k vertices makes the graph interval). Other parameters could be obtained by analyzing the hardness proof, or analyzing the data or the dimension. We refer to the survey of Niedermeier [57] for more detailed exposition on this. Bodlaender and Jansen [47] parameterized p -VERTEX COVER by the size of a feedback vertex set. The reason for this parameterization of the p -VERTEX COVER is interesting because the minimum size of a feedback vertex is always at most the size of the vertex cover number. It was shown in [47] that this parameterized problem admits a cubic kernel. See [11,12,47,48] for other studies of kernelization for parameterizing one problem by the solution to the other problem. Parameterizing a graph optimization problem with other graph optimization problem like vertex cover number, max-leaf number have been studied before from the algorithmic perspective [32] but so far there are very few results from the view point of kernelization complexity.

\mathcal{F} -Deletion problem. Let \mathcal{F} be a finite set of graphs. In an p - \mathcal{F} -DELETION problem, we are given an n -vertex graph G and an integer k as input, and asked whether at most k vertices can be deleted from G such that the resulting graph does not contain a graph from \mathcal{F} as a minor. We refer to such subset as \mathcal{F} -hitting set. The p - \mathcal{F} -DELETION problem is a generalization of several fundamental problems. For example, when $\mathcal{F} = \{K_2\}$, a complete graph on two vertices, this is p -VERTEX COVER. When $\mathcal{F} = \{C_3\}$, a cycle on three vertices, this is the p -FEEDBACK VERTEX SET problem. It is known that p -VERTEX COVER and p -FEEDBACK VERTEX SET admit polynomial kernels [17,60]. It was shown in [36] that when \mathcal{F} is a graph with two vertices connected by constant number of parallel edges, then p - \mathcal{F} -DELETION also admits a polynomial kernel. Recently, it has been shown that p - \mathcal{F} -DELETION admits a polynomial kernel whenever \mathcal{F} contains a planar graph [35]. This generalizes several results in the area including for p -VERTEX COVER, p -FEEDBACK VERTEX SET and p -PATHWIDTH

1-DELETION. Finally, an interesting direction for further research here is to investigate p - \mathcal{F} -DELETION when none of the graphs in \mathcal{F} is planar. The most interesting case here is when $\mathcal{F} = \{K_5, K_{3,3}\}$ aka the VERTEX PLANARIZATION problem. Surprisingly, we are not aware even of a single case of p - \mathcal{F} -DELETION with \mathcal{F} containing no planar graph admitting a polynomial kernel.

Kernelization Lower Bounds. It is known that p -LEAF OUT-BRANCHING admits n independent kernels of size $O(k^3)$ [33]. It is not a kernel in the usual “many to one” sense, but it is a kernel in the “one to many” sense. We can generalize the notion of many to one kernels to Turing kernelization. In order to define this we first define the notion of t -oracle.

Definition 9. A t -oracle for a parameterized problem Π is an oracle that takes as input (I, k) with $|I| \leq t$, $k \leq t$ and decides whether $(I, k) \in \Pi$ in constant time.

Definition 10. A parameterized problem Π is said to have $g(k)$ -sized turing kernel if there is an algorithm which given an input (I, k) together with a $g(k)$ -oracle for Π decides whether $(I, k) \in \Pi$ in time polynomial in $|I|$ and k . $|x'|, k' \leq g(k)$.

Observe that both the well known notion of kernels and many to one kernels are special cases of turing kernelization. In particular, many to one kernels are equivalent to turing kernels where the kernelization algorithm is only allowed to make one oracle call and must return the same answer as the oracle.

Problem 1. Is there a framework to rule out the possibility of having one to many or Turing kernels similar to the framework developed in [9,38]?

Problem 2. Which other problems admit a Turing kernelization like the quadratic kernels for k -LEAF OUT-BRANCHING and k -LEAF OUT-TREE? Does the problem of finding a path of length at most k admit a Turing kernel (even on planar graphs)?

Problem 3. Does there exist a problem for which we do not have a linear many-to-one kernel, but does have linear kernels from the viewpoint of Turing kernelization?

Recently, there has been an attempt to answer the first question in [46] by organizing problems into complexity classes which are closed under polynomial parameter transformations. It is shown that many of the problems which are known not to have polynomial kernels unless $CoNP \subseteq NP/poly$ are equivalent with respect to Turing kernels. Specifically, either all of them have Turing kernels or all of them do not. The problems belonging to this class include CONNECTED VERTEX COVER and MIN ONES SAT. Interestingly, LONGEST PATH is not shown to belong to this class, leaving some hope that the problem might have a Turing kernel.

We conclude the survey with the following concrete open problem.

Problem 4. Does p -DIRECTED FEEDBACK VERTEX SET admit a polynomial kernel?

References

1. Abu-Khzam, F.N.: A kernelization algorithm for d -hitting set. *J. Comput. Syst. Sci.* 76(7), 524–531 (2010)
2. Alber, J., Fellows, M.R., Niedermeier, R.: Polynomial-time data reduction for dominating set. *Journal of the ACM* 51(3), 363–384 (2004)
3. Alon, N., Gutin, G., Kim, E.J., Szeider, S., Yeo, A.: Solving MAX- r -SAT above a tight lower bound. In: *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2010)*, pp. 511–517. SIAM (2010)
4. Alon, N., Gutin, G., Krivelevich, M.: Algorithms with large domination ratio. *J. Algorithms* 50, 118–131 (2004)
5. Alon, N., Yuster, R., Zwick, U.: Color-coding. *J. Assoc. Comput. Mach.* 42(4), 844–856 (1995)
6. Arnborg, S., Courcelle, B., Proskurowski, A., Seese, D.: An algebraic theory of graph reduction. *J. ACM* 40(5), 1134–1164 (1993)
7. Bodlaender, H., Fomin, F.V., Lokshtanov, D., Penninkx, E., Saurabh, S., Thilikos, D.M.: (Meta) Kernelization. In: *Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2009)*, pp. 629–638. IEEE (2009)
8. Bodlaender, H.L., de Fluiter, B.: Reduction Algorithms for Constructing Solutions in Graphs with Small Treewidth. In: Cai, J.-Y., Wong, C.K. (eds.) *COCOON 1996*. LNCS, vol. 1090, pp. 199–208. Springer, Heidelberg (1996)
9. Bodlaender, H.L., Downey, R.G., Fellows, M.R., Hermelin, D.: On problems without polynomial kernels. *J. Comput. Syst. Sci.* 75(8), 423–434 (2009)
10. Bodlaender, H.L., Hagerup, T.: Parallel algorithms with optimal speedup for bounded treewidth. *SIAM J. Comput.* 27, 1725–1746 (1998)
11. Bodlaender, H.L., Jansen, B.M.P., Kratsch, S.: Cross-composition: A new technique for kernelization lower bounds. In: *Proceedings of the 28th International Symposium on Theoretical Aspects of Computer Science (STACS 2011)*. LIPIcs, vol. 9, pp. 165–176. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2011)
12. Bodlaender, H.L., Jansen, B.M.P., Kratsch, S.: Preprocessing for Treewidth: A Combinatorial Analysis through Kernelization. In: Aceto, L., Henzinger, M., Sgall, J. (eds.) *ICALP 2011, Part I*. LNCS, vol. 6755, pp. 437–448. Springer, Heidelberg (2011)
13. Bodlaender, H.L., Thomassé, S., Yeo, A.: Analysis of data reduction: Transformations give evidence for non-existence of polynomial kernels, Tech. Report CS-UU-2008-030, Department of Information and Computer Sciences, Utrecht University, Utrecht, The Netherlands (2008)
14. Bodlaender, H.L., van Antwerpen-de Fluiter, B.: Reduction algorithms for graphs of small treewidth. *Inf. Comput.* 167(2), 86–119 (2001)
15. Bourgain, J.: Walsh subspaces of l^p -product space. *Seminar on Functional Analysis, Exp. (4A)*, 9 (1980)
16. Burrage, K., Estivill-Castro, V., Fellows, M.R., Langston, M.A., Mac, S., Rosamond, F.A.: The Undirected Feedback Vertex Set Problem Has a Poly(k) Kernel. In: Bodlaender, H.L., Langston, M.A. (eds.) *IWPEC 2006*. LNCS, vol. 4169, pp. 192–202. Springer, Heidelberg (2006)
17. Chen, J., Kanj, I.A., Jia, W.: Vertex cover: further observations and further improvements. *Journal of Algorithms* 41, 280–301 (2001)

18. Chor, B., Fellows, M., Juedes, D.W.: Linear Kernels in Linear Time, or How to Save k Colors in $o(n^2)$ Steps. In: Hromkovič, J., Nagl, M., Westfechtel, B. (eds.) WG 2004. LNCS, vol. 3353, pp. 257–269. Springer, Heidelberg (2004)
19. Crowston, R., Gutin, G., Jones, M., Kim, E.J., Ruzsa, I.Z.: Systems of Linear Equations over \mathbb{F}_2 and Problems Parameterized above Average. In: Kaplan, H. (ed.) SWAT 2010. LNCS, vol. 6139, pp. 164–175. Springer, Heidelberg (2010)
20. Cygan, M., Kratsch, S., Pilipeczuk, M., Pilipeczuk, M., Wahlström, M.: Clique cover and graph separation: New incompressibility results. CoRR, abs/1111.0570 (2011)
21. de Fluiter, B.: Algorithms for Graphs of Small Treewidth. PhD thesis, Utrecht University (1997)
22. Dell, H., Marx, D.: Kernelization of packing problems. In: SODA, pp. 68–81 (2012)
23. Dell, H., van Melkebeek, D.: Satisfiability allows no nontrivial sparsification unless the polynomial-time hierarchy collapses. In: STOC, pp. 251–260 (2010)
24. Diestel, R.: Graph theory, 3rd edn. Graduate Texts in Mathematics, vol. 173. Springer, Berlin (2005)
25. Dom, M., Guo, J., Hüffner, F., Niedermeier, R., Truß, A.: Fixed-Parameter Tractability Results for Feedback Set Problems in Tournaments. In: Calamoneri, T., Finocchi, I., Italiano, G.F. (eds.) CIAC 2006. LNCS, vol. 3998, pp. 320–331. Springer, Heidelberg (2006)
26. Downey, R.G., Fellows, M.R.: Parameterized complexity. Springer, Heidelberg (1999)
27. Downey, R.G., Fellows, M.R., Stege, U.: Computational tractability: the view from Mars. Bull. Eur. Assoc. Theor. Comput. Sci. EATCS (69), 73–97 (1999)
28. Drucker, A.: On the hardness of compressing an AND of SAT instances, Theory Lunch, February 17, Center for Computational Intractability (2012), <http://intractability.princeton.edu/blog/2012/03/theory-lunch-february-17/>
29. Erdős, P., Rado, R.: Intersection theorems for systems of sets. J. London Math. Soc. 35, 85–90 (1960)
30. Fellows, M.R.: The Lost Continent of Polynomial Time: Preprocessing and Kernelization. In: Bodlaender, H.L., Langston, M.A. (eds.) IWPEC 2006. LNCS, vol. 4169, pp. 276–277. Springer, Heidelberg (2006)
31. Fellows, M.R., Langston, M.A.: An analogue of the myhill-nerode theorem and its use in computing finite-basis characterizations (extended abstract). In: FOCS, pp. 520–525 (1989)
32. Fellows, M.R., Lokshtanov, D., Misra, N., Mnich, M., Rosamond, F.A., Saurabh, S.: The complexity ecology of parameters: An illustration using bounded max leaf number. Theory Comput. Syst. 45(4), 822–848 (2009)
33. Fernau, H., Fomin, F.V., Lokshtanov, D., Raible, D., Saurabh, S., Villanger, Y.: Kernel(s) for problems with no kernel: On out-trees with many leaves. In: STACS 2009, pp. 421–432. Schloss Dagstuhl—Leibniz-Zentrum fuer Informatik (2009)
34. Flum, J., Grohe, M.: Parameterized Complexity Theory. Texts in Theoretical Computer Science, An EATCS Series, Springer, Berlin (2006)
35. Fomin, F., Lokshtanov, D., Misra, N., Saurabh, S.: Planar- \mathcal{F} Deletion: Approximation, Kernelization and Optimal FPT algorithms (2012) (unpublished manuscript)
36. Fomin, F.V., Lokshtanov, D., Misra, N., Philip, G., Saurabh, S.: Hitting forbidden minors: Approximation and kernelization. In: Proceedings of the 28th International Symposium on Theoretical Aspects of Computer Science (STACS 2011). LIPIcs, vol. 9, pp. 189–200. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2011)

37. Fomin, F.V., Lokshantov, D., Saurabh, S., Thilikos, D.M.: Bidimensionality and kernels. In: Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2010), pp. 503–510. SIAM (2010)
38. Fortnow, L., Santhanam, R.: Infeasibility of instance compression and succinct PCPs for NP. In: STOC 2008: Proceedings of the 40th Annual ACM Symposium on Theory of Computing, pp. 133–142. ACM (2008)
39. Garey, M.R., Johnson, D.S.: Computers and Intractability; A Guide to the Theory of NP-Completeness. W. H. Freeman & Co., New York (1990)
40. Gramm, J., Guo, J., Hüffner, F., Niedermeier, R.: Data reduction and exact algorithms for clique cover. *ACM Journal of Experimental Algorithmics* 13 (2008)
41. Guo, J., Niedermeier, R.: Invitation to data reduction and problem kernelization. *SIGACT News* 38, 31–45 (2007)
42. Gutin, G., Kim, E.J., Szeider, S., Yeo, A.: A probabilistic approach to problems parameterized above or below tight bounds. *J. Comput. Syst. Sci.* 77, 422–429 (2011)
43. Gutin, G., van Iersel, L., Mnich, M., Yeo, A.: All Ternary Permutation Constraint Satisfaction Problems Parameterized above Average Have Kernels with Quadratic Numbers of Variables. In: de Berg, M., Meyer, U. (eds.) *ESA 2010, Part I. LNCS*, vol. 6346, pp. 326–337. Springer, Heidelberg (2010)
44. Hall, P.: On representatives of subsets. *J. London Math. Soc.* 10, 26–30 (1935)
45. Håstad, J., Venkatesh, S.: On the advantage over a random assignment. In: Proceedings of the 34th Annual ACM Symposium on Theory of Computing (STOC 2002), pp. 43–52. ACM (2002)
46. Hermelin, D., Kratsch, S., Soltys, K., Wahlström, M., Wu, X.: Hierarchies of inefficient kernelizability. *CoRR*, abs/1110.0976 (2011)
47. Jansen, B.M.P., Bodlaender, H.L.: Vertex cover kernelization revisited: Upper and lower bounds for a refined parameter. In: Proceedings of the 28th International Symposium on Theoretical Aspects of Computer Science (STACS 2011). *LIPICs*, vol. 9, pp. 177–188. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2011)
48. Jansen, B.M.P., Kratsch, S.: Data Reduction for Graph Coloring Problems. In: Owe, O., Steffen, M., Telle, J.A. (eds.) *FCT 2011. LNCS*, vol. 6914, pp. 90–101. Springer, Heidelberg (2011)
49. König, D.: Über Graphen und ihre Anwendung auf Determinantentheorie und Mengenlehre. *Math. Ann.* 77, 453–465 (1916)
50. Kratsch, S.: Co-nondeterminism in compositions: a kernelization lower bound for a ramsey-type problem. In: SODA, pp. 114–122 (2012)
51. Kratsch, S., Wahlström, M.: Representative sets and irrelevant vertices: New tools for kernelization. *CoRR*, abs/1111.2195 (2011)
52. Kratsch, S., Wahlström, M.: Compression via matroids: a randomized polynomial kernel for odd cycle transversal. In: SODA, pp. 94–103 (2012)
53. Lokshantov, D.: Phd thesis, *New Methods in Parameterized Algorithms and Complexity* (2009)
54. Mahajan, M., Raman, V.: Parameterizing above guaranteed values: Maxsat and maxcut. *J. Algorithms* 31(2), 335–354 (1999)
55. Misra, N., Raman, V., Saurabh, S.: Lower bounds on kernelization. *Discrete Optim.* 8, 110–128 (2011)
56. Niedermeier, R.: Invitation to fixed-parameter algorithms. *Oxford Lecture Series in Mathematics and its Applications*, vol. 31. Oxford University Press, Oxford (2006)

57. Niedermeier, R.: Reflections on multivariate algorithmics and problem parameterization. In: Proceedings of the 27th International Symposium on Theoretical Aspects of Computer Science (STACS 2010). LIPIcs, vol. 5, pp. 17–32. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2010)
58. Quine, W.V.: The problem of simplifying truth functions. *Amer. Math. Monthly* 59, 521–531 (1952)
59. Stockmeyer, L.J.: The polynomial-time hierarchy. *Theor. Comp. Sc.* 3, 1–22 (1976)
60. Thomassé, S.: A quadratic kernel for feedback vertex set. *ACM Transactions on Algorithms* 6 (2010)

Parameterized Complexity and Subexponential-Time Computability*

Jianer Chen^{1,**} and Iyad A. Kanj^{2,***}

¹ Department of Computer Science and Engineering, Texas A&M University,
College Station, TX 77843

chen@cs.tamu.edu

² School of Computing, DePaul University, 243 S. Wabash Avenue,
Chicago, IL 60604-2301

ikanj@cs.depaul.edu

Abstract. Since its inception in the 1990's, parameterized complexity has established itself as one of the major research areas in theoretical computer science. Parameterized and kernelization algorithms have proved to be very useful for solving important problems in various domains of science and technology. Moreover, parameterized complexity has shown deep connections to traditional areas of theoretical computer science, such as structural complexity theory and approximation algorithms.

In this paper, we discuss some of the recent results pertaining to the relation between parameterized complexity and subexponential-time computability. We focus our attention on satisfiability problems because they play a key role in the definition of both parameterized complexity and structural complexity classes, and because they model numerous important problems in computer science.

1 Introduction

Parameterized complexity was established in the early 1990's by the seminal work of Downey and Fellows. It was instigated by the demands of real-world applications, and by the belief that computational complexity should “serve the community,” and should be “used not only in the pursuit of the declared objectives but also in the design of heuristic and approximation algorithms for

* This paper is dedicated to the 60th birthday of Michael R. Fellows. Many of the results examined in this paper were authored or co-authored by Michael, and those that were not, would probably never have existed without his efforts. If parameterized complexity would not have started without Michael, Rodney, and two bottles of 1989 Villa Maria Merlot/Cabernet Sauvignon, then it definitely would not have flourished and matured into such an important area of theoretical computer science without Michael's great ideas, efforts, and inspirations.

** This work was supported in part by the USA National Science Foundation under the grants CCF-0830455 and CCF-0917288.

*** Supported in part by a DePaul University Competitive Research Grant.

problems that are hard, but for which, nevertheless, something must be done” [24]. Today, applications of efficient parameterized and kernelization algorithms for solving important problems that are otherwise intractable from the traditional complexity theory perspective, are prevalent. Parameterized complexity has matured into a very exciting research area of theoretical computer science, with applications spanning numerous domains of science and technology.

The rich positive toolkit of novel techniques for designing efficient parameterized and kernelization algorithms was accompanied by a corresponding “negative” toolkit that supports a theory of parameterized intractability. While studying the parameterized intractability of the DOMINATING SET problem [24], Downey and Fellows sharply observed that there is a rich structure in parameterized intractability theory. A parameterized intractability hierarchy, the W -hierarchy, was subsequently introduced to classify the level of intrinsic intractability of parameterized problems [24]. Research in the last twenty years revealed that the W -hierarchy provides a deep structural characterization of parameterized complexity [24]. In addition, the theory is remarkably applicable to a wide range of natural computational problems. This motivated researchers in theoretical computer science, and in parameterized complexity in particular, to investigate the structural relation between parameterized complexity and traditional areas of theoretical computer science (e.g., structural complexity, approximation, etc).

Perhaps one of the most important problems that has deep roots in the aforementioned areas is the satisfiability problem, including all its variants. The currently-best algorithms for satisfiability problems are essentially based on brute-force methods that enumerate all possible solutions, which obviously requires exponential time. It has become clear that the existence of faster exponential-time algorithms for satisfiability problems is closely related to the computational intractability of a large class of well-known NP-hard problems, measured from a number of different angles, such as computational time and space, fixed-parameter tractability, and approximation. For example, Impagliazzo, Paturi, and Zane showed that the subexponential-time computability of the 3-SAT problem is equivalent to the subexponential time computability of a large class of well-known NP-hard problems [31]; this class is closed under subexponential-time preserving reductions, called *serf-reductions*. This led researchers in theoretical computer science to formulate a hypothesis, which became known as the *exponential-time hypothesis*, shortly ETH, conjecturing that no member in this class is solvable in subexponential time.

The subexponential-time computability of weighted satisfiability on bounded depth circuits is closely related to the fixed-parameter tractability of the W -hierarchy in parameterized complexity theory (see for instance [1, 9, 12, 14, 19–21, 23, 24]). Research on parameterized complexity revealed more subtle relations between the computational complexity of NP-hard problems and the (sub)exponential-time computability of satisfiability problems [9, 12, 14, 33, 39]. For example, it is now known that efficient polynomial-time approximation schemes of a number of NP-hard problems, and parameterized algorithms that

are asymptotically more efficient than the brute-force enumeration for many W -hard problems, are all dependent of the subexponential-time computability of various satisfiability problems (see for instance [9, 12, 14, 33, 39]). In particular, due to the aforementioned research, it is now known that the classification of parameterized intractability has a correspondence to the exact computability of satisfiability problems on various circuit families. This line of research deepened our understanding of the structural relation between the two computational frameworks of parameterized complexity and subexponential-time computability, and resulted in new tools for deriving computational lower bounds on parameterized computation, exact computation, and approximation algorithms.

In the current paper, we discuss some of the results related to the relation between the parameterized intractability and the computational complexity of a variety of satisfiability problems. This relation is not surprising, given that the W -hierarchy in parameterized complexity was defined mainly based on weighted satisfiability problems. Nevertheless, the close connection between these two different research frameworks has not been rigorously studied until very recently.

The systematic research in this direction started with the results of Abrahamson, Downey, and Fellows [1], and Downey and Fellows [24]. In [1], it was shown that $W[P] = FPT$ ($W[P]$ is the parameterized complexity class characterized by the weighted satisfiability problem restricted to polynomial-size Boolean circuits) implies that CIRCUIT SATISFIABILITY (satisfiability of polynomial-size circuits) is computable in subexponential time. It was also shown in [1] that, for any even $t \geq 1$, the collapse of the W -hierarchy at its t -th level (i.e., $W[t] = FPT$) implies that SAT[t] (t -level satisfiability) is solvable in time $2^{o(n)}m^{O(1)}$ (m is the instance size).¹ This later result was refined by Downey and Fellows [24] to all levels of the W -hierarchy. Downey and Fellows [24] showed that: (1) for any $t \geq 2$, $W[t] = FPT$ implies that SAT[t] is computable in subexponential time; and (2) $W[1] = FPT$ implies that ETH fails, which subsequently implies the subexponential-time computability of several well-known problems including 3-SAT, INDEPENDENT SET, and VERTEX COVER. Those results were further refined and extended in [12, 14], where it was shown that the condition $W[t] = FPT$ in (1) and (2) can be relaxed (see Section 4), and that the subexponential-time computability of SAT[t], in turn, implies the collapse of the W -hierarchy at its $(t - 1)$ -st level ($W[t - 1] = FPT$) for $t \geq 2$, and for $t = 1$ implies the failure of ETH. The previous results were exploited further in [12, 14] to derive lower bounds on the computability and the approximation of well-known NP-hard problems, such as INDEPENDENT SET, CLIQUE, DOMINATING SET, based on parameterized complexity hypotheses. Important questions along this line of research remained open however, including the following: What is the equivalent, from the parameterized complexity perspective, of the subexponential-time computability of various satisfiability problems?

¹ The $o(\cdot)$ notation in this paper denotes the $o^{\text{eff}}(\cdot)$ notation (see, for instance, [26]). More formally, by writing $f(n) = o(g(n))$ we mean that there exists a computable nondecreasing unbounded function $\mu(n) : \mathbb{N} \rightarrow \mathbb{N}$, and $n_0 \in \mathbb{N}$, such that $f(n) \leq g(n)/\mu(n)$ for all $n \geq n_0$.

Downey et al. [23] were the first to try to answer this question. They defined a parameterized complexity class, called $M[1]$, comprised between FPT and $W[1]$, consisting of a “miniaturization” of weighted circuit satisfiability, and showed that $M[1] = FPT$ is equivalent to ETH fails. The idea of a miniaturization of a problem was explored earlier in the work of Abrahamson et al. [1], and Cai and Juedes [9], and Downey et al. [23] provided a formal definition for this notion and studied it systematically. Flum and Grohe [25], Chen and Flum [19, 20], and Chen and Grohe [21], launched a systematic study of the relation between parameterized complexity and subexponential-time computability, using the notion of miniaturization. Chen and Grohe [21] were able to give a correspondence between the subexponential-time computability of certain satisfiability problems and parameterized complexity classes.

We will focus on some of the key results pertaining to the relation between parameterized complexity and subexponential-time computability, and their applications. We will also try to describe some of the problems that remain open in this line of research. While we tried our best to include most of the recent results on to these topics, we do apologize in advance for any relevant result that we may have omitted; certainly, this was not our intention.

Before we close this section, we would like to mention a recent breakthrough-result in complexity theory by Williams [42], who proved that the non-uniform ACC class does not contain $NTIME[2^n]$, the class of languages that are solvable in nondeterministic time $O(2^n)$. This is regarded as a very significant advance in complexity theory, as it was even unknown whether the class EXP^{NP} is contained in a weaker ACC class of languages accepted by circuit families of polynomial-size and depth 3 with more restricted modular gates. There are at least two directions in which Williams’ result is relevant to parameterized complexity and subexponential-time computability. First, a major component of Williams’ approach is faster exact algorithms for satisfiability problems. In particular, Williams developed a $2^{n-\Omega(n^\delta)}$ -time algorithm for the satisfiability problem on subexponential-size ACC-circuits, where δ is a constant dependent on the circuit depth. Second, the computation model considered, i.e., ACC-circuits, is closely related to the generic complete problems for the W -hierarchy, i.e., the weighted satisfiability on bounded depth Boolean circuits [24].

2 Preliminaries

A *circuit* is a directed acyclic graph. The nodes of in-degree 0 are called *inputs*, and are labeled either by *positive literals* x_i or by *negative literals* \bar{x}_i . The nodes of in-degree larger than 0 are called *gates* and are labeled with Boolean operators AND or OR. A special gate of out-degree 0 is designated as the *output* node. We do not allow NOT gates in the above circuit model, since by De Morgan’s laws, a general circuit can be effectively converted into the above circuit model. A circuit is said to be *monotone* (resp. *antimonotone*) if all its input literals are positive (resp. negative). The *depth* of a circuit is the maximum distance from an input node to the output gate of the circuit. A circuit represents a Boolean function in

a natural way. Using the results in [11], every circuit can be re-structured into an equivalent circuit with the same monotonicity and number of input variables, same depth, and such that all inputs are in level 0, all AND and OR gates are organized into alternating levels with edges only going from a level to the next level, and with at most a polynomial increase in the circuit size. Thus, without loss of generality, we will implicitly assume that circuits are in this leveled form. A circuit is a Π -circuit if its output gate is an AND gate, and is a Π_h -circuit if it is a Π -circuit of depth h . We say that a truth assignment τ to the input variables of a circuit C satisfies a gate g in C if τ makes the gate g have value 1, and that τ satisfies the circuit C if τ satisfies the output gate of C . A circuit C is satisfiable if there is a truth assignment to the input variables of C that satisfies C . The weight of an assignment τ is the number of variables assigned value 1 by τ . A CNF formula is a conjunction of a set of clauses where each clause is a disjunction of literals. For a CNF formula F with n input variables, we can naturally correspond an equivalent Π_2 -circuit C_F with n input variables.

A parameterized problem Q is a subset of $\Omega^* \times \mathbb{N}$, where Ω is a fixed alphabet and \mathbb{N} is the set of all non-negative integers. Each instance of the parameterized problem Q is a pair (x, k) , where the second component, i.e., the non-negative integer k , is called the parameter. We say that the parameterized problem Q is fixed-parameter tractable [24] if there is a (parameterized) algorithm that decides whether an input (x, k) is a member of Q in time $f(k)|x|^c$, where c is a fixed constant and $f(k)$ is a computable function independent of the input length $|x|$. Let FPT denote the class of all fixed-parameter tractable parameterized problems.

The Π_t -CIRCUIT SATISFIABILITY problem where $t \geq 2$, abbreviated SAT[t] henceforth, is defined as follows: Given a Π_t -circuit C , decide if C is satisfiable. For instance, the SAT[2] problem is the same as the satisfiability problem on CNF formulas (CNF-SAT). We will also study the parameterized problems based on the “weighted version” of the satisfiability problems on circuits. In particular, for $t \geq 2$, the WEIGHTED Π_t -CIRCUIT SATISFIABILITY problem, abbreviated WCS[t] is for a given Π_t -circuit C and a given parameter k , to decide if C has a satisfying assignment of weight k . Similarly, the WEIGHTED MONOTONE Π_t -CIRCUIT SATISFIABILITY problem, abbreviated WCS⁺[t], and the WEIGHTED ANTIMONOTONE Π_t -CIRCUIT SATISFIABILITY problem, abbreviated WCS⁻[t] are the WCS[t] problems on, respectively, monotone circuits and antimonotone circuits. We denote by WCNF 2-SAT⁻ the WCS⁻[2] problem with the restriction that the fan-in of each gate at level 1 of the input circuit is bounded by 2. Equivalently, each instance of WCNF 2-SAT⁻ consists of a parameter k and a CNF formula in which all literals are negative and each clause is a disjunction of at most two literals. Finally, let 3-SAT be the CNF-SAT problem with the restriction that each clause in the input formula is a disjunction of at least 3 literals.

The optimization class SNP introduced by Papadimitriou and Yannakakis [38] consists of all search problems expressible by second-order existential formulas whose first-order part is universal. Impagliazzo and Paturi [31] introduced the notion of completeness for the class SNP under *serf-reductions*, and identified a

class of problems which are complete for SNP under serf-reductions, such that the subexponential-time computability for any of these problems implies the subexponential-time computability of all problems in SNP. Many well-known NP-hard problems are proved to be complete for SNP under the serf-reduction, including 3-SAT, VERTEX COVER, and INDEPENDENT SET, for which extensive efforts have been made in the last three decades to develop subexponential-time algorithms with no success [43]. This fact has led to the *exponential-time hypothesis*, ETH, which is equivalent to the statement that not all SNP problems are solvable in subexponential-time:

Exponential-Time Hypothesis (ETH): The problem 3-SAT cannot be solved in time $2^{o(n)}$, where n is the number of variables in the input formula.

The ETH has become a standard hypothesis in the area of parameterized algorithms and complexity and of exact algorithms and complexity.

To study the fixed-parameter tractability, the *fpt-reduction* has been introduced [24]: a parameterized problem Q is *fpt-reducible* to a parameterized problem Q' if there is an algorithm M that transforms each instance (x, k) of Q into an instance $(x', g(k))$ (g is a function of k only) of Q' in time $f(k)|x|^c$, where f and g are computable functions and c is a constant, such that $(x, k) \in Q$ if and only if $(x', g(k)) \in Q'$.

Based on the notion of fpt-reducibility, a hierarchy of parameterized complexity, the *W-hierarchy*, has been introduced. At the 0-th level of the hierarchy lies the class *FPT*, and at the i -th level for $i > 0$, the class $W[i]$. The original definition of the $W[i]$ classes was based on the notion of the *weft* of a circuit, which is the maximum number of “large gates” (i.e., gates whose fan-in is larger than a prespecified constant) on any path from an input gate to the output gate of the circuit [24]. The previous definition, however, was shown to be equivalent to the following definition by Downey and Fellows (see [24]), which we use in this paper. The class $W[1]$ consists of all parameterized problems that are fpt-reducible to the problem WCNF 2-SAT⁻. For an even $t \geq 2$, the class $W[t]$ consists of all parameterized problems that are fpt-reducible to the problem WCS⁺[t], and for an odd $t \geq 3$, the class $W[t]$ consists of all parameterized problems that are fpt-reducible to the problem WCS⁻[t]. To simplify our statements, we will denote by WCS^{*}[t] the problem WCS⁺[t] if t is even and the problem WCS⁻[t] if t is odd. Therefore, for all $t \geq 2$, the class $W[t]$ consists of all parameterized problems that are fpt-reducible to the problem WCS^{*}[t].

A parameterized problem Q is *W[i]-hard* if every problem in $W[i]$ is fpt-reducible to Q , and is *W[i]-complete* if in addition Q is in $W[i]$. By the definition, the problem WCNF 2-SAT⁻ is $W[1]$ -complete, and for all $t \geq 2$, the problem WCS^{*}[t] is $W[t]$ -complete. If any $W[i]$ -hard problem is in *FPT*, then $W[i] = \text{FPT}$, which, to the common belief of researchers in parameterized complexity, is very unlikely [24].

We have the following relation among parameterized complexity classes [24]:

$$\text{FPT} \subseteq W[1] \subseteq W[2] \subseteq \dots$$

For more information about parameterized complexity we refer the reader to [24, 26, 36].

3 ETH, $W[1]$, and CNF-SAT

In this section we discuss some of the results on the relation between the parameterized complexity and the subexponential-time computability of important satisfiability problems on Π_2 -circuits, and their consequences on the computability of some natural NP-hard problems. We start with the following folklore results that can be easily verified by the reader (see for example [9]):

Lemma 1. *If a parameterized problem is solvable in time $2^{o(k \log n)} n^{O(1)}$, where n is the input length and k is the parameter, then the problem is fixed-parameter tractable.*

Fact 31. *The function $(c \log n)^{O(k)}$ is bounded above by $f(k)n^{O(1)}$, where f is a function of k only.*

The following result is a consequence of a result in [9]:

Theorem 1. *If CNF-SAT is solvable in time $2^{o(n)} m^{O(1)}$, then $W[1] = FPT$, where n is the number of variables and m is the formula size.*

Proof. Suppose that CNF-SAT is solvable in time $2^{o(n)} m^{O(1)}$. We consider the WCNF 2-SAT⁻ problem. Since this problem is complete for $W[1]$, it suffices to show that it can be solved in time $f(k)m_1^{O(1)}$ where f is a function independent of the circuit size m_1 . Note that since $m_1 = n^{O(1)}$, where n is the number of variables in the circuit (each gate at level-1 has fan-in at most 2), the problem reduces to showing that WCNF 2-SAT⁻ is solvable in time $f(k)n^{O(1)}$.

Let (C, k) be an instance of WCNF 2-SAT⁻, and note that the gates at level 1 in C are OR gates, each of fan-in at most two. Let $\bar{x}_1, \dots, \bar{x}_n$ be the input literals to C . We will construct a circuit C' from C with $k \lceil \log n \rceil$ input variables, such that C has a weight- k assignment if and only if C' is satisfiable. The input variables in C' are divided into k blocks B_1, \dots, B_k , where block B_i , $i = 1, \dots, k$, consists of $r = \lceil \log n \rceil$ input variables z_i^1, \dots, z_i^r . Also, for every input variable z_i^j , $i \in \{1, \dots, k\}$, $j \in \{1, \dots, r\}$, we associate the input literal \bar{z}_i^j to denote its negation. Informally speaking, each block B_i will contain the encoding of an input variable whose value is 1 in a weight- k assignment to C . We show how to connect the new input variables and their negations to the level-1 OR gates in C . Let g be a gate at level-1 in C , and suppose that \bar{x}_p, \bar{x}_q , are connected to g . (We assume that g has fan-in exactly two as the case when g has fan-in 1 is much easier to handle.) Now \bar{x}_p is 1 if and only if x_p is 0, if and only if none of the blocks B_i , $i = 1, \dots, k$ contains the binary representation of p . Thus, in C' we will connect the new input variables to g as follows. We introduce k new OR gates g_p^1, \dots, g_p^k . Each gate g_p^i , $i = 1, \dots, k$, has exactly r inputs, and its input comes only from input variables in block B_i and their negations. Informally speaking, each gate g_p^i will be satisfied

if and only if block B_i does not contain the binary representation of p , and hence, does not encode x_p . Suppose that the binary representation of p is $b_1 b_2 \dots b_r$. For $i = 1, \dots, k$, the input to g_p^i is determined as follows. For $j = 1, \dots, r$, if $b_j = 0$, then connect z_i^j to g_p^i , and if $b_j = 1$, then connect \bar{z}_i^j to g_p^i . Now replace the connection from \bar{x}_p to g by the connections from all gates g_p^i , $i = 1, \dots, k$ to an AND gate g_p which feeds into g . We do the same for \bar{x}_q . Now gate g is equivalent to $g_p \vee g_q$, where $g_p = \bigwedge_{i=1}^k g_p^i$ and $g_q = \bigwedge_{i=1}^k g_q^i$. By the distributive law, we can write $g = \bigwedge_{i,j=1,\dots,k} (g_p^i \vee g_q^j)$, and the AND gate in g can be merged with the output AND gate of the circuit C . We repeat the above construction for every level-1 gate in C . Since the g_p^i 's and the g_q^j 's for every level-1 gate g are OR gates, the resulting circuit is a two-level circuit, where the top level consists of OR gates that all feed into the single output AND gate of C .

Now we can add enforcement circuitry to ensure that the k blocks encode k distinct input variables. This can be simply achieved by adding a circuitry consisting of $\binom{k}{2}$ subcircuits, each subcircuit enforces that the two blocks that feed into it are distinct. To do so, each subcircuit performs a bitwise XOR operation to the corresponding variables in the two blocks. Since the number of input variables to each subcircuit is $O(\log n)$, each subcircuit can be transformed into a subcircuit in the CNF form in $n^{O(1)}$ time, whose output AND gate can then be merged with the output AND gate of C .

Let C' be the resulting circuit. Clearly, C' has size $n^{O(1)}$ and can be constructed in $n^{O(1)}$ time. Moreover, from the above discussion, we know that C' consists of two levels, where the top level consists only of OR gates, and the bottom level consists of the output AND gate of the circuit. Since the k input blocks in C' basically encode the k input variables in C with value 1 in a weight- k assignment to C , it is not difficult to verify that C has a weight- k truth assignment if and only if C' is satisfiable. Now C' is an instance of CNF-SAT with $k \cdot r$ input variables. It follows that we can decide if C' is satisfiable in time bounded by $2^{o(k \log n)} n^{O(1)}$. By Lemma 1, it follows that WCNF 2-SAT^- is fixed-parameter tractable.

Using reductions to problem kernel, and some standard self reductions, Cai and Juedes [9] were able to preclude the existence of subexponential-time parameterized algorithms for several problems under the assumption that ETH holds (n is the instance size, and k is the natural parameter in the corresponding problem):

Theorem 2 ([9]). *The following problems can be solved in time $2^{o(k)} n^{O(1)}$ if and only if ETH fails: VERTEX COVER, MAX h -SAT, Δ -VERTEX COVER (the graph has degree bounded by the constant Δ), Δ -INDEPENDENT SET, and Δ -DOMINATING SET.*

Note that all the above problems can be solved in time $2^{O(k)} n^{O(1)}$. So assuming ETH, the above theorem rules out the existence of significantly-better parameterized algorithms for those problems than the ones that are currently known.

Using kernelization, planarization techniques, and standard reductions, Cai and Juedes [9] were able to extend the above lower bound results to planar graphs:

Theorem 3 ([9]). *Unless ETH fails, the following problems cannot be solved in time $2^{o(\sqrt{k})}n^{O(1)}$: PLANAR VERTEX COVER, PLANAR INDEPENDENT SET, PLANAR DOMINATING SET, and PLANAR RED/BLUE DOMINATING SET.*

Some of the results in the previous theorem can also be extended to bounded-degree planar graphs [17] to obtain the same lower bounds.

The following theorem can be viewed as providing a partial converse to Theorem 1, after noting that the statement that ETH fails is equivalent to subexponential-time computability of 3-SAT. This result is due to Downey and Fellows [24]. The proof given here, which appears in [12], is different than the original proof, since the original proof was a corollary of a more general result.

Theorem 4. *If $W[1] = FPT$ then the hypothesis ETH fails.*

Proof. Since INDEPENDENT SET is $W[1]$ -complete under the fpt-reduction [24] and VERTEX COVER is complete for SNP under self-reductions [31], it suffices to show that if INDEPENDENT SET is solvable in time $f(k)n^{O(1)}$ then VERTEX COVER is solvable in time $2^{o(k)}n^{O(1)}$. Assume that there is an algorithm A which determines whether there exists an independent set of size k in a graph G with n vertices in $f(k)n^{O(1)}$ time. We will show that the VERTEX COVER problem can be solved in time $2^{o(k)}n^{O(1)}$. Without loss of generality, we can assume that the function f is nondecreasing, unbounded, and that $f(k) \geq 2^k$. Define f^{-1} by $f^{-1}(h) = \max\{q \mid f(q) \leq h\}$. Since the function f is nondecreasing and unbounded, the function f^{-1} is also nondecreasing and unbounded, and satisfies $f(f^{-1}(h)) \leq h$. From $f(k) \geq 2^k$, we have $f^{-1}(h) \leq \log h$.

Let $(G = (V, E), k)$ be an instance of VERTEX COVER. By the kernelization result for VERTEX COVER [16], we can assume that G has at most $n \leq 2k$ vertices. We partition the n vertices of G into $k' = \lfloor f^{-1}(k) \rfloor$ blocks $B_1, B_2, \dots, B_{k'}$ each of size at most $\lceil \frac{n}{\lfloor f^{-1}(k) \rfloor} \rceil$. (Without loss of generality, we shall assume that $\lfloor f^{-1}(k) \rfloor \geq 1$.) Observe that G has a vertex cover of size k if and only if there exists a way to partition k into $k_1, \dots, k_{k'}$ (i.e., $k = k_1 + k_2 + \dots + k_{k'}$), and there are subsets $V'_i \subseteq B_i$, $i = 1, \dots, k'$ with $|V'_i| = k_i$, such that $\bigcup_{i=1}^{k'} V'_i$ is a vertex cover for G . Since $|B_i| \leq \lceil \frac{n}{\lfloor f^{-1}(k) \rfloor} \rceil$, this approach converts the single question “does G have a vertex cover of size k ?” into at most

$$\begin{aligned} \left(\lceil \frac{n}{\lfloor f^{-1}(k) \rfloor} \rceil\right)^{k'} &\leq \left(\lceil \frac{2k}{\lfloor f^{-1}(k) \rfloor} \rceil\right)^{\lfloor f^{-1}(k) \rfloor} \\ &\leq (2k)^{f^{-1}(k)} \\ &\leq 2^{\log(2k) \cdot f^{-1}(k)} \\ &\leq 2^{\log(2k) \cdot \log k} = 2^{o(k)} \end{aligned}$$

more restrictive questions of the type “does G have a vertex cover V' of size $k = k_1 + k_2 + \dots + k_{k'}$ with $|B_i \cap V'| = k_i$?”. Hence, we can determine whether G has a vertex cover of size k by answering at most $2^{o(k)}$ questions individually.

To answer each of the $2^{o(k)}$ questions, we use the algorithm A for INDEPENDENT SET. Given G , k , and $k_1, \dots, k_{k'}$ such that $k = k_1 + k_2 + \dots + k_{k'}$, we construct a graph $G^* = (V^*, E^*)$ as follows. For each block of vertices B_i in G , and for each subset $B_{ij} \subseteq B_i$ with $|B_{ij}| = k_i$, add a vertex v_{ij} to V^* if B_{ij} is a vertex cover of $G(B_i)$ (the subgraph of G induced by B_i). Add edges to E^* so that the collection of the vertices v_{ij} associated with block B_i , $i = 1, \dots, k'$, forms a clique. In addition, for each $v_{ij}, v_{kl} \in V^*$, where $i \neq k$, add the edge (v_{ij}, v_{kl}) to E^* if $B_{ij} \cup B_{kl}$ does not form a vertex cover for $G(B_i \cup B_k)$. This completes the construction of G^* . To determine if G has a vertex cover of size k with the properties mentioned above, it suffices to use algorithm A to determine if G^* has an independent set of size k' . We prove the correctness of this claim.

Assume that G^* has an independent set I of size k' . Since G^* has k' disjoint cliques, exactly one vertex from each set $V_i^* = \{v_{ij} \mid v_{ij} \in V^*\}$ is in I . Let $V' = \cup_{v_{ij} \in I} B_{ij}$. Since $|B_{ij}| = k_i$, and at most one B_{ij} is included in V' , it follows that $|V' \cap B_i| = k_i$, and $|V'| = k$. Thus, it suffices to prove that V' is a vertex cover of G . Let $(u, v) \in E$, and let $u \in B_i$ and $v \in B_k$. If $i = k$, then it must be the case that either u or $v \in V'$. To see this, note that there exists a $v_{ij} \in I \subseteq V^*$, which means that $B_{ij} \subseteq V'$ by the definition of V' . Since $v_{ij} \in V^*$, B_{ij} is a vertex cover of $G(B_i)$, and either u or v must be in $B_{ij} \subseteq V'$. Suppose now that $i \neq k$, and let v_{ij}, v_{kl} be the two vertices in V_i^* and V_k^* , respectively, that are in I . Then it must be the case that $u \in B_{ij}$ or $v \in B_{kl}$, otherwise $B_{ij} \cup B_{kl}$ is not a vertex cover of $G(B_i \cup B_k)$, which would imply that there is an edge between v_{ij} and v_{kl} in G^* , contradicting the fact that I is an independent set of G^* . It follows that either u or v is in V' . This shows that V' is a vertex cover of G . To prove the converse, assume that G has a vertex cover V' of size $k = k_1 + k_2 + \dots + k_{k'}$ with $|B_i \cap V'| = k_i$. Let $I = \{v_{ij} \mid B_{ij} = B_i \cap V'\}$. It is clear that $I \subseteq V^*$ and $|I| = k'$, since for each i , B_{ij} has k_i vertices and it is a vertex cover of $G(B_i)$. Furthermore, I is an independent set in G^* because for each $v_{ij}, v_{kl} \in I$, $(v_{ij}, v_{kl}) \notin E^*$. This is true since $B_{ij} \cup B_{kl} = V' \cap (B_i \cup B_k)$ is a vertex cover of $G(B_i \cup B_k)$.

Therefore, we can use algorithm A to determine whether G has a vertex cover V' of size $k = k_1 + k_2 + \dots + k_{k'}$, by checking whether G^* has an independent set I of size k' . The graph G^* has at most $N = 2^{\lceil \frac{2k}{f^{-1}(k)} \rceil} \cdot k' \leq 2^{\lceil \frac{2k}{f^{-1}(k)} \rceil} \cdot f^{-1}(k) = 2^{o(k)}$ vertices because $|B_i| \leq \lceil \frac{2k}{f^{-1}(k)} \rceil$, and there are at most $2^{|B_i|}$ possible subsets B_{ij} of size k_i . Therefore, the time taken by applying the algorithm A to the instance (G^*, k') is of the order

$$f(k')N^{O(1)} \leq f(f^{-1}(k))N^{O(1)} \leq k \cdot N^{O(1)} = 2^{o(k)}n^{O(1)}$$

after observing that $N^{O(1)} = 2^{o(k)}$. Noting that the time needed to construct G^* is $N^{O(1)} = 2^{o(k)}$, and that applying the kernelization algorithm for VERTEX COVER takes polynomial time in n , it follows that the VERTEX COVER problem can be solved in time $n^{O(1)} + 2^{o(k)} \cdot 2^{o(k)} \cdot n^{O(1)} = 2^{o(k)}n^{O(1)}$. This completes the proof.

In fact, Theorem 4 can be strengthened to the following result:

Theorem 5. *If the $W[1]$ -complete problem $WCNF\ 2-SAT^-$ is solvable in time $f(k)m^{o(k)}$, where m is the size of the input formula, then the hypothesis ETH fails.*

The interested readers are referred to [12, 14] for a detailed proof.

4 A General Framework

In this section, we present two generic results that establish certain relations between the parameterized complexity of weighted satisfiability problems and the subexponential-time computability of their unweighted versions. Since weighted satisfiability problems are complete for the W -hierarchy, this leads to a relation between the subexponential-time computability of natural satisfiability problems and the collapse of the W -hierarchy. In Section 6, we will present some applications of these results to obtain computational lower bounds on the parameterized complexity and on the approximation of natural problems. The results are mainly due to the work in [12, 14], and can be viewed as generalizations and strengthening of the results in the previous section.

We start with the following lemma, which will be used in the proof of the next theorem:

Lemma 2. *Let $t \geq 2$ be an integer. There is an algorithm A_1 that, for a given integer $r > 0$, transforms each Π_t -circuit C_1 of n_1 input variables and size m_1 into an instance (C_2, k) of $WCS^*[t]$, where $k = \lceil n_1/r \rceil$ and the Π_t -circuit C_2 has $n_2 = 2^r k$ input variables and size $m_2 \leq 2m_1 + 2^{2r+1}k$, such that C_1 is satisfiable if and only if (C_2, k) is a yes-instance of $WCS^*[t]$. The running time of the algorithm A_1 is bounded by $O(m_2^2)$.*

Proof. Let $k = \lceil n_1/r \rceil$. Divide the n_1 input variables x_1, \dots, x_{n_1} of the Π_t -circuit C_1 into k blocks B_1, \dots, B_k , where block B_i consists of input variables $x_{(i-1)r+1}, \dots, x_{ir}$, for $i = 1, \dots, k - 1$, and block B_k consists of input variables $x_{(k-1)r+1}, \dots, x_{n_1}$. Denote by $|B_i|$ the number of variables in block B_i . Then $|B_i| = r$, for $1 \leq i \leq k - 1$, and $|B_k| \leq r$. For an integer j , $0 \leq j \leq 2^{|B_i|} - 1$, denote by $\text{bin}_i(j)$ the length- $|B_i|$ binary representation of j , which can also be interpreted as an assignment to the variables in block B_i .

We construct a new set of input variables in k blocks B'_1, \dots, B'_k . Each block B'_i consists of $s = 2^r$ variables $z_{i,0}, z_{i,1}, \dots, z_{i,s-1}$. The Π_t -circuit C_2 is constructed from the Π_t -circuit C_1 by replacing the input gates in C_1 by the new input variables in B'_1, \dots, B'_k . We consider two cases.

Case 1. t is even. Then all level-1 gates in the Π_t -circuit C_1 are OR gates. We connect the new variables $z_{i,j}$ to these level-1 gates to construct the circuit C_2 as follows. Let x_q be an input variable in C_1 such that x_q is the h -th variable in block B_i . If the positive literal x_q is an input to a level-1 OR gate g_1 in C_1 , then all positive literals $z_{i,j}$ in block B'_i such that $0 \leq j \leq 2^{|B_i|} - 1$ and the h -th bit in $\text{bin}_i(j)$ is 1 are connected to gate g_1 in the circuit C_2 . If the negative literal

\bar{x}_q is an input to a level-1 OR gate g_2 in C_1 , then all positive literals $z_{i,j}$ in block B'_i such that $0 \leq j \leq 2^{|B_i|} - 1$ and the h -th bit in $\text{bin}_i(j)$ is 0 are connected to gate g_2 in the circuit C_2 .

Note that if the size $|B_k|$ of the last block B_k in C_1 is smaller than r , then the above construction for block B'_k is only on the first $2^{|B_k|}$ variables in B'_k , and the last $s - 2^{|B_k|}$ variables in B'_k have no output edges, and hence become “dummy variables”.

We also add an “enforcement” circuitry to the circuit C_2 to ensure that every satisfying assignment to C_2 assigns the value 1 to at least one variable in each block B'_i . This can be achieved by having an OR gate for each block B'_i , whose inputs are connected to all positive literals in block B'_i and whose output is an input to the output gate of the circuit C_2 (for block B'_k , the inputs of the OR gate are from the first $2^{|B_k|}$ variables in B'_k). This completes the construction of the circuit C_2 . It is easy to see that the circuit C_2 is a monotone Π_t -circuit (note that $t \geq 2$ and hence the enforcement circuitry does not increase the depth of C_2). Thus, (C_2, k) is an instance of the problem $\text{WCS}^+[t]$.

We verify that the circuit C_1 is satisfiable if and only if the circuit C_2 has a satisfying assignment of weight k . Suppose that the circuit C_1 is satisfied by an assignment τ . Let τ_i be the restriction of τ to block B_i , $1 \leq i \leq k$. Let j_i be the integer such that $\text{bin}_i(j_i) = \tau_i$. Then according to the construction of the circuit C_2 , by setting $z_{i,j_i} = 1$ and all other variables in B'_i to 0, we can satisfy all level-1 OR gates in C_2 whose corresponding level-1 OR gates in C_1 are satisfied by the assignment τ_i . Doing this for all blocks B_i , $1 \leq i \leq k$, gives a weight- k assignment τ' to the circuit C_2 that satisfies all level-1 OR gates in C_2 whose corresponding level-1 OR gates in C_1 are satisfied by τ . Since τ satisfies the circuit C_1 , the weight- k assignment τ' satisfies the circuit C_2 .

Conversely, suppose that the circuit C_2 is satisfied by a weight- k assignment τ' . Because of the enforcement circuitry in C_2 , τ' assigns the value 1 to exactly one variable in each block B'_i (in particular, in block B'_k , this variable must be one of the first $2^{|B_k|}$ variables in B'_k). Now suppose that in block B'_i , τ' assigns the value 1 to the variable z_{i,j_i} . Then we set an assignment τ_i to the block B_i in C_1 such that $\tau_i = \text{bin}_i(j_i)$. By the construction of the circuit C_2 , the level-1 OR gates satisfied by the variable $z_{i,j_i} = 1$ are all satisfied by the assignment τ_i . Therefore, if we make an assignment τ to the circuit C_1 such that the restriction of τ to block B_i is τ_i for all i , then the assignment τ will satisfy all level-1 OR gates in C_1 whose corresponding level-1 OR gates in C_2 are satisfied by τ' . Since τ' satisfies the circuit C_2 , we conclude that the circuit C_1 is satisfiable.

This completes the proof that when t is even, the circuit C_1 is satisfiable if and only if the constructed pair (C_2, k) is a yes-instance of $\text{WCS}^+[t]$.

Case 2. t is odd. Then all level-1 gates in the Π_t -circuit C_1 are AND gates. We connect the new variables $z_{i,j}$ to these level-1 gates to construct the circuit C_2 as follows. Let x_q be an input variable in C_1 and be the h -th variable in block B_i . If the positive literal x_q is an input to a level-1 AND gate g_1 in C_1 , then all negative literals $\bar{z}_{i,j}$ in block B'_i such that $0 \leq j \leq 2^{|B_i|} - 1$ and the h -th bit in $\text{bin}_i(j)$ is 0 are inputs to gate g_1 in C_2 . If the negative literal \bar{x}_q is an input to

a level-1 AND gate g_2 in C_1 , then all negative literals $\bar{z}_{i,j}$ in block B'_i such that $0 \leq j \leq 2^{|B_i|} - 1$ and the h -th bit in $\text{bin}_i(j)$ is 1 are inputs to gate g_2 in C_2 .

For the last $s - 2^{|B_k|}$ variables in the last block B'_k in C_2 , we connect the negative literals $\bar{z}_{k,j}$, $2^{|B_k|} \leq j \leq s - 1$, to the output gate of the circuit C_2 (thus, the variables $z_{k,j}$, $2^{|B_k|} \leq j \leq s - 1$, are forced to have the value 0 in any satisfying assignment to C_2).

An enforcement circuitry is added to C_2 to ensure that every satisfying assignment to C_2 assigns the value 1 to at most one variable in each block B'_i . This can be achieved as follows. For every two distinct negative literals $\bar{z}_{i,j}$ and $\bar{z}_{i,h}$ in B'_i , $0 \leq j, h \leq 2^{|B_i|} - 1$, add an OR gate $g_{j,h}$. Connect $\bar{z}_{i,j}$ and $\bar{z}_{i,h}$ to $g_{j,h}$ and connect $g_{j,h}$ to the output AND gate of C_2 . This completes the construction of the circuit C_2 . The circuit C_2 is an antimonotone Π_t -circuit (again the enforcement circuitry does not increase the depth of C_2). Thus, (C_2, k) is an instance of the problem $\text{WCS}^- [t]$.

We verify that the circuit C_1 is satisfiable if and only if the circuit C_2 has a satisfying assignment of weight k . Suppose that the circuit C_1 is satisfied by an assignment τ . Let τ_i be the restriction of τ to block B_i , $1 \leq i \leq k$. Let j_i be the integer such that $\text{bin}_i(j_i) = \tau_i$. Consider the weight- k assignment τ' to C_2 that for each i assigns $z_{i,j_i} = 1$ and all other variables in B'_i to 0. We show that τ' satisfies the circuit C_2 . Let g_1 be a level-1 AND gate in C_1 that is satisfied by the assignment τ . Since C_2 is antimonotone, all inputs to g_1 in C_2 are negative literals. Since all negative literals except \bar{z}_{i,j_i} in block B'_i have the value 1, we only have to prove that no \bar{z}_{i,j_i} from any block B'_i is an input to g_1 . Assume to the contrary that \bar{z}_{i,j_i} in block B'_i is an input to g_1 . Then by the construction of the circuit C_2 , there is a variable x_q that is the h -th variable in block B_i such that either x_q is an input to g_1 in C_1 and the h -th bit of $\text{bin}_i(j_i)$ is 0, or \bar{x}_q is an input to g_1 in C_1 and the h -th bit of $\text{bin}_i(j_i)$ is 1. However, by our construction of the index j_i from the assignment τ , if the h -th bit of $\text{bin}_i(j_i)$ is 0 then τ assigns $x_q = 0$, and if the h -th bit of $\text{bin}_i(j_i)$ is 1 then τ assigns $x_q = 1$. In either case, τ would not satisfy the gate g_1 , contradicting our assumption. Thus, for all i , no \bar{z}_{i,j_i} is an input to the gate g_1 , and the assignment τ' satisfies the gate g_1 . Since g_1 is an arbitrary level-1 AND gate in C_2 , we conclude that the assignment τ' satisfies all level-1 AND gates in C_2 whose corresponding gates in C_1 are satisfied by the assignment τ . Since τ satisfies the circuit C_1 , the weight- k assignment τ' satisfies the circuit C_2 .

Conversely, suppose that the circuit C_2 is satisfied by a weight- k assignment τ' . Because of the enforcement circuitry in C_2 , the assignment τ' assigns the value 1 to exactly one variable in each block B'_i (in particular, this variable in block B'_k must be one of the first $2^{|B_k|}$ variables in B'_k since the last $s - 2^{|B_k|}$ variables in B'_k are forced to have the value 0 in the satisfying assignment τ'). Suppose that in block B'_i , τ' assigns the value 1 to the variable z_{i,j_i} . Then we set an assignment $\tau_i = \text{bin}_i(j_i)$ to block B_i in C_1 . Let τ be the assignment whose restriction on block B_i is τ_i . We prove that τ satisfies the circuit C_1 . In effect, if a level-1 AND gate g_2 in C_2 is satisfied by the assignment τ' , then no negative literal \bar{z}_{i,j_i} is an input to g_2 . Suppose that g_2 is not satisfied by τ in C_1 , then either a

positive literal x_q is an input to g_2 and τ assigns $x_q = 0$, or a negative literal \overline{x}_q is an input to g_2 and τ assigns $x_q = 1$. Let x_q be the h -th variable in block B_i . If τ assigns $x_q = 0$ then the h -th bit in $\text{bin}_i(j_i)$ is 0. Thus, x_q cannot be an input to g_2 in C_1 because otherwise by our construction the negative literal \overline{x}_{i,j_i} would be an input to g_2 in C_2 . On the other hand, if τ assigns $x_q = 1$ then the h -th bit in $\text{bin}_i(j_i)$ is 1, thus, \overline{x}_q cannot be an input to g_2 in C_1 because otherwise the negative literal \overline{x}_{i,j_i} would be an input to g_2 in C_2 . This contradiction shows that the gate g_2 must be satisfied by the assignment τ . Since g_2 is an arbitrary level-1 AND gate in C_2 , we conclude that the assignment τ satisfies all level-1 AND gates in C_1 whose corresponding level-1 AND gates in C_2 are satisfied by the assignment τ' . Since τ' satisfies the circuit C_2 , the assignment τ satisfies the circuit C_1 and hence the circuit C_1 is satisfiable.

This completes the proof that when t is odd, the Π_t -circuit C_1 is satisfiable if and only if the pair (C_2, k) is a yes-instance of $\text{WCS}^-[t]$.

Summarizing the above discussion, we conclude that for any $t \geq 2$, from a Π_t -circuit C_1 of n_1 input variables and size m_1 , we can construct an instance (C_2, k) of the problem $\text{WCS}^*[t]$ such that C_1 is satisfiable if and only if (C_2, k) is a yes-instance of $\text{WCS}^*[t]$. Here $k = \lceil n_1/r \rceil$, and C_2 has $n_2 = 2^r k$ input variables and size $m_2 \leq m_1 + n_2 + k + k2^{2r} \leq 2m_1 + k2^{2r+1}$ (where the term $k + k2^{2r}$ is an upper bound on the size of the enforcement circuitry). Finally, it is straightforward to verify that the pair (C_2, k) can be constructed from the circuit C_1 in time $O(m_2^2)$.

Theorem 6. *Let $t \geq 2$ be an integer. For any function f , if the problem $\text{WCS}^*[t]$ is solvable in time $f(k)n^{o(k)}m^{O(1)}$, then the problem $\text{SAT}[t]$ can be solved in time $2^{o(n)}m^{O(1)}$.*

Proof. Suppose that there is an algorithm M_{WCS} of running time bounded by $f(k)n^{k/\lambda(k)}p(m)$ that solves the problem $\text{WCS}^*[t]$, where $\lambda(k)$ is a nondecreasing and unbounded function and p is a polynomial. Without loss of generality, we can assume that the function f is nondecreasing, unbounded, and that $f(k) \geq 2^k$. Define f^{-1} by $f^{-1}(h) = \max\{q \mid f(q) \leq h\}$. Since the function f is nondecreasing and unbounded, the function f^{-1} is also nondecreasing and unbounded, and satisfies $f(f^{-1}(h)) \leq h$. From $f(k) \geq 2^k$, we have $f^{-1}(h) \leq \log h$.

Now we solve the problem $\text{SAT}[t]$ as follows. For an instance C_1 of $\text{SAT}[t]$, where C_1 is a Π_t -circuit of n_1 input variables and size m_1 , we set the integer $r = \lfloor 3n_1/f^{-1}(n_1) \rfloor$, and call the algorithm A_1 in Lemma 2 to convert C_1 into an instance (C_2, k) of the problem $\text{WCS}^*[t]$. Here $k = \lceil n_1/r \rceil$, C_2 is a Π_t -circuit of $n_2 = 2^r k$ input variables and size $m_2 \leq 2m_1 + 2^{2r+1}k$, and the algorithm A_1 takes time $O(m_2^2)$. According to Lemma 2, we can determine if C_1 is a yes-instance of $\text{SAT}[t]$ by calling the algorithm M_{WCS} to determine if (C_2, k) is a yes-instance of $\text{WCS}^*[t]$. The running time of the algorithm M_{WCS} on (C_2, k) is bounded by $f(k)n_2^{k/\lambda(k)}p(m_2)$. Combining all above we get an algorithm M_{SAT} of running time $f(k)n_2^{k/\lambda(k)}p(m_2) + O(m_2^2)$ for the problem $\text{SAT}[t]$. We analyze the running time of the algorithm M_{SAT} in terms of the values n_1 and m_1 .

Since $k = \lceil n_1/r \rceil \leq f^{-1}(n_1) \leq \log n_1$,² we have $f(k) \leq f(f^{-1}(n_1)) \leq n_1$. Moreover,

$$k = \lceil n_1/r \rceil \geq n_1/r \geq n_1/(3n_1/f^{-1}(n_1)) = f^{-1}(n_1)/3.$$

Therefore if we set $\lambda'(n_1) = \lambda(f^{-1}(n_1)/3)$, then $\lambda(k) \geq \lambda'(n_1)$. Since both λ and f^{-1} are nondecreasing and unbounded, $\lambda'(n_1)$ is a nondecreasing and unbounded function of n_1 . We have (note that $k \leq f^{-1}(n_1) \leq \log n_1$),

$$\begin{aligned} n_2^{k/\lambda(k)} &= (k2^r)^{k/\lambda(k)} \leq k^k 2^{kr/\lambda(k)} \leq k^k 2^{3kn_1/(\lambda(k)f^{-1}(n_1))} \leq k^k 2^{3n_1/\lambda(k)} \\ &\leq k^k 2^{3n_1/\lambda'(n_1)} = 2^{o(n_1)}. \end{aligned}$$

Finally, consider the factor m_2 . Since f^{-1} is nondecreasing and unbounded,

$$m_2 \leq 2m_1 + k2^{2r+1} \leq 2m_1 + 2 \log n_1 2^{6n_1/f^{-1}(n_1)} = 2^{o(n_1)}m_1.$$

Therefore, both terms $p(m_2)$ and $O(m_2^2)$ in the running time of the algorithm M_{Sat} are bounded by $2^{o(n_1)}p'(m_1)$ for a polynomial p' . Combining all these, we conclude that the running time $f(k)n_2^{k/\lambda(k)}p(m_2) + O(m_2^2)$ of M_{Sat} is bounded by $2^{o(n_1)}p'(m_1)$ for a polynomial p' . Hence, the problem $\text{SAT}[t]$ can be solved in time $2^{o(n)}m^{O(1)}$. This completes the proof of the theorem.

The following corollary follows directly from the above theorem, and can be seen as a generalization of Theorem 4 to higher levels of the W -hierarchy and the satisfiability problem. This result is due to Abrahamson et al. [1], and to Downey and Fellows [24]:

Corollary 1. *Let $t \geq 2$ be an integer. If $W[t] = FPT$ then the problem $\text{SAT}[t]$ can be solved in time $2^{o(n)}m^{O(1)}$.*

In fact, Theorem 6 remains valid even if we restrict the parameter values to be bounded by an arbitrarily small function, as shown in the following theorem, whose proof is omitted and can be found in [14]:

Theorem 7. *Let $t \geq 2$ be an integer, and $\mu(n)$ a nondecreasing and unbounded function. If for a function f , the problem $\text{WCS}^*[t]$ is solvable in time $f(k)n^{o(k)}m^{O(1)}$ for parameter values $k \leq \mu(n)$, then the problem $\text{SAT}[t]$ can be solved in time $2^{o(n)}m^{O(1)}$.*

The following theorem can be viewed as a generalization of Theorem 1:

Theorem 8. *For any $t \geq 2$, if $\text{SAT}[t]$ can be solved in time $2^{o(n)}h(m)$ for some polynomial h , then $W[t-1] = FPT$.*

² Without loss of generality, we assume that in our discussions, all values under the ceiling function “ $\lceil \cdot \rceil$ ” and the floor function “ $\lfloor \cdot \rfloor$ ” are greater than or equal to 1. Therefore, we will always assume the inequalities $\lceil \beta \rceil \leq 2\beta$ and $\lfloor \beta \rfloor \geq \beta/2$ for any value β .

Proof. If $t = 2$, the theorem states that if CNF-SAT can be solved in time $2^{o(n)}h(m)$ then $W[1] = FPT$. This is basically the result in Theorem 1, which was proved in the previous section. Thus, we can assume that $t \geq 3$. Suppose that $SAT[t]$ is solvable in time $2^{o(n)}h(m)$. Then there exists an unbounded non-decreasing function $s(n)$ such that $SAT[t]$ can be solved in time bounded by $2^{n/s(n)}h(m)$. We distinguish two cases based on the parity of t .

Case 1. t is odd. We consider the $WCS^+[t - 1]$ problem. Since this problem is complete for $W[t - 1]$, it suffices to show that this problem can be solved in time $f(k)h'(m)$ where f is a function independent of the circuit size m , and h' is a polynomial. Let (C, k) be an instance of $WCS^+[t - 1]$, where C has n input variables and size m . Since $t - 1$ is even, the gates at level 1 in C are OR gates. Let x_1, \dots, x_n be the input variables to C . We will construct a circuit C' from C with $k \lceil \log n \rceil$ input variables, such that C has a weight k assignment if and only if C' is satisfiable. The input variables in C' are divided into k blocks B_1, \dots, B_k , where block B_i , $i = 1, \dots, k$, consists of $r = \lceil \log n \rceil$ input variables z_i^1, \dots, z_i^r . Also, for every input variable z_i^j , $i \in \{1, \dots, k\}$, $j \in \{1, \dots, r\}$, we associate the input literal \bar{z}_i^j to denote its negation. Informally speaking, each block B_i will contain the encoding of an input variable whose value is 1 in a weight- k assignment to C . We show how to connect the new input variables and their negations to the level-1 OR gates in C . Let g be a level-1 OR gate in C . Let x_p be an input to g , and let $b_1 b_2 \dots b_r$ be the binary representation of the number p (if there are fewer than r bits in the binary representation of p , we pad the binary representation of p with the appropriate number of 0's on the left to make it consist of exactly r bits). We introduce k new AND gates g_p^1, \dots, g_p^k . Each gate g_p^i , $i = 1, \dots, k$, has exactly r inputs, and its input comes only from input variables in block B_i and their negations. Informally speaking, each gate g_p^i will be satisfied if and only if block B_i contains the binary representation of p , and hence, encodes x_p . The input to gate g_p^i is determined as follows. For $j = 1, \dots, r$, if $b_j = 0$, then connect \bar{z}_i^j to g_p^i , and if $b_j = 1$, then connect z_i^j to g_p^i . Now replace the connection from x_p to g by the connections from all gates g_p^i , $i = 1, \dots, k$, to g . We repeat this process for every level-1 gate g in C and every input variable in $\{x_1, \dots, x_n\}$ to g . Clearly, this construction only adds a single level to the circuit C consisting of AND gates, and hence, the resulting circuit is a Π_t circuit. We also add enforcement circuitry to ensure that the k blocks B_i , $i = 1, \dots, k$, encode distinct k variables. This can be simply achieved by adding a circuitry that performs a bitwise XOR operation to the corresponding variables in every two blocks, which can be done by adding a 3-level AND-of-OR-of-AND subcircuits to every two blocks (note that the last AND can be merged with the output AND gate of the circuit if $t = 3$). Clearly, the resulting circuit is still a Π_t -circuit. Moreover, the size of C is only increased by a polynomial factor in its original size. Let C'_F be the circuit resulting from this construction. From the above discussion we know that C' is a Π_t -circuit of size $h'(m)$ for some polynomial h' . Since the k input blocks in C' basically encode the k input variables in C with value 1 in a weight- k assignment to C , it is not difficult to verify that C has a weight- k truth assignment if

and only if C' is satisfiable. Now C' is an instance of $\text{SAT}[t]$ with kr input variables. It follows that we can decide if C' is satisfiable in time bounded by $T(n) = 2^{kr/s(kr)}h(h'(m)) = 2^{k\lceil \log n \rceil / s(k\lceil \log n \rceil)}h(h'(m)) \leq 2^{k(\log n + 1)/s'(n)}h''(m)$, for some unbounded non-decreasing function $s'(n)$, and some polynomial h'' . Thus $T(n) \in 2^{o(\log n)k}h''(m)$, and $\text{WCS}^+[t-1]$ is solvable in time $2^{o(\log n)k}h''(m)$ for some polynomial h'' . It follows that $\text{WCS}^+[t-1]$ is fixed parameter tractable (see Lemma 1), and hence, $W[t-1] = \text{FPT}$.

Case 2. t is even, and hence $t-1 \geq 3$ is odd. We consider the $\text{WCS}^-[t-1]$ problem, which is complete for $W[t-1]$. The proof proceeds in a very similar fashion to the proof of **Case 1** above. Let (C, k) be an instance of $\text{WCS}^-[t-1]$, and note that the gates at level 1 in C are AND gates. Let $\bar{x}_1, \dots, \bar{x}_n$ be the input literals to C , and let r and $B_i, i = 1, \dots, k$, be as defined above. Again, block B_i will be used to encode the indices of the input variables in C that are set to 1 in a weight- k assignment to C . Let g be a gate at level-1 in C , and suppose that \bar{x}_p , where $p \in \{1, \dots, n\}$, is connected to g . Now \bar{x}_p is 1 if and only if x_p is 0, if and only if none of the blocks $B_i, i = 1, \dots, k$ contains the binary representation of p . Thus, in C' we will connect the new input variables to g as follows. We introduce k new OR gates g_p^1, \dots, g_p^k . Each gate $g_p^i, i = 1, \dots, k$, has exactly r inputs, and its input comes only from input variables in block B_i and their negations. Informally speaking, each gate g_p^i will be satisfied if and only if block B_i does not contain the binary representation of p , and hence, does not encode x_p . Suppose the binary representation of p is $b_1b_2 \dots b_r$. For $i = 1, \dots, k$, the input to g_p^i is determined as follows. For $j = 1, \dots, r$, if $b_j = 0$, then connect z_i^j to g_p^i , and if $b_j = 1$, then connect \bar{z}_i^j to g_p^i . Now replace the connection from \bar{x}_p to g by the connections from all gates $g_p^i, i = 1, \dots, k$ to g , and repeat that for every level-1 gate in C and every original input literal to that gate. This adds an OR-level to C , thus increasing the number of levels in C by 1, and resulting in a Π_t -circuit. Now we can add the enforcement circuitry to ensure that all k blocks encode k distinct input variables. This can be simply achieved by adding a circuitry that performs a bitwise XOR operation to the corresponding variables in every two blocks. The resulting circuitry that tests that no two blocks are the same can be implemented by an OR-of-AND-of-AND-of-OR subcircuit (the last AND gate can be identified with the output gate of C if $t = 4$). Since $t \geq 4$, the resulting circuit C' is a Π_t -circuit whose size is not more than a polynomial in the size of C . The proof from this point on proceeds in exactly the same fashion as in **Case 1** above.

It follows that $W[t-1] = \text{FPT}$. This completes the proof.

5 The Miniaturization Classes

As we have seen in the previous section, the subexponential-time computability of the satisfiability problem on circuits of depth t , for $t \geq 2$, implies the fixed-parameter tractability of the class $W[t-1]$, and is implied by the fixed-parameter tractability of the class $W[t]$. Also, the failure of the ETH hypothesis

is implied by the fixed-parameter tractability of the class $W[1]$. One may naturally ask whether the subexponential-time computability of these satisfiability problems, and also of other important NP-hard problems, is *equivalent* to the fixed-parameter tractability of some parameterized problems. In particular, is the hypothesis ETH *equivalent* to the fixed-parameter intractability of a particular parameterized problem?

The problem was initially considered in [9]. Downey *et al.* [23] formally proposed a process, named *parameterized miniaturization*, that establishes the equivalence between the subexponential-time computability of problems and the fixed-parameter tractability of their corresponding miniaturized parameterized problems. To describe this process, we need to be more careful in the use of the “size” of problem instances. Note that this had not been a problem for polynomial-time computation because any reasonable choice of instance size is polynomially related to the length of a reasonable encoding of the instance, and polynomial-time computation is robust for these variations. On the other hand, when we study subexponential-time computation, we implicitly allow only linear changes in the metric based on which the complexity of the computation is measured. Therefore, we have to be very careful in the use of certain conventional metrics for instance size, such as the number of Boolean variables in a satisfiability problem and the number of vertices in a graph. For example, if we use the number n of variables as a metric in the CNF-SAT problem, then CNF-SAT is certainly not solvable in subexponential-time in terms of n (i.e., in time $2^{o(n)}$) because the length of any encoding of an instance can be $2^{\Omega(n)}$. On the other hand, if we use the length l of a binary encoding of an instance of n variables and m clauses of CNF-SAT as the metric, then, because $l \geq (n+m) \log n$, CNF-SAT can be obviously solved in subexponential-time in l : a simple brute-force algorithm takes time $O(2^{nm} \log n) = 2^{o(l)}$.

In the following discussion, we shall assume that we use a “natural” size for the problem instances. In particular, the size of a circuit will be the number of input variables plus the number of links in the circuit, the size of a Boolean formula will be the number of occurrences of literals in the formula, and the size of a graph is the number of its vertices plus the number of its edges.

We first consider the 3-SAT problem in terms of the size s of the input formula. The miniaturization process of the 3-SAT problem gives the following parameterized problem:

MINI(3-SAT):

Given nonnegative integers m and k in unary, and an instance F of size bounded by $k \log m$ for 3-SAT, where k is the parameter, decide if F is satisfiable.

The following theorem follows from a similar proof by Downey *et al.* [23] for circuit satisfiability:

Theorem 9 ([23]). *The parameterized problem MINI(3-SAT) is fixed parameter tractable if and only if the 3-SAT problem is solvable in subexponential time, i.e., in time $2^{o(s)}$, where s is the formula size.*

Proof. Suppose that 3-SAT is solvable in time $2^{o(s)}$ by an algorithm \mathcal{A} . Given an instance (F, m, k) of MINI(3-SAT), where the Boolean formula F has size bounded by $s = k \log m$, we simply invoke the algorithm \mathcal{A} on the formula F to decide, in time $2^{o(s)} = 2^{o(k \log m)}$, whether F is satisfiable or not. By Lemma 1, this shows that MINI(3-SAT) is fixed-parameter tractable.

Conversely, suppose that MINI(3-SAT) is fixed parameter tractable, and hence is solvable by an algorithm \mathcal{A}' in time $f(k)|x|^{O(1)}$, where f is a computable function of k , and $|x| = O(km)$ is the length of (any reasonable encoding of) the instance $x = (F, m, k)$ of MINI(3-SAT). Without loss of generality, we can assume that $f(k) \geq k$ for all k and that f is a strictly increasing function, from which we derive that the inverse function f^{-1} is well-defined and unbounded, and satisfies the condition $f^{-1}(n) \leq n$. For an instance F' of size s for the 3-SAT problem, let $k = f^{-1}(s)$ and $m = 2^{s/k}$, and consider the instance $x = (F', m, k)$ of MINI(3-SAT) (note that the size of F' is $s = k \log m$, and that $|x| = O(m + k + s \log s) = O(2^{s/f^{-1}(s)}s^2)$). By invoking the algorithm \mathcal{A}' on (F', m, k) , we can decide whether F' is satisfiable or not in time

$$f(k)|x|^{O(1)} = s^{O(1)}2^{O(s/f^{-1}(s))} = 2^{o(s)},$$

where the last equality holds true because f^{-1} is a non-decreasing and unbounded function. This proves that 3-SAT can be solved in time $2^{o(s)}$, and hence, completes the proof of the theorem.

The 3-SAT problem measured by formula size is complete for the class SNP under serf-reductions, in the sense that if 3-SAT is solvable in subexponential time, then all problems in SNP are solvable in exponential time [30]. As mentioned before, there is a large number of important NP-hard problems that are complete for the class SNP under the serf-reduction, including the 3-SAT problem measured by *the number of input variables*. Therefore, the ETH hypothesis is equivalent to the statement that the 3-SAT problem measured by formula size is not subexponential-time solvable. This, combined with Theorem 9, gives us:

Theorem 10 ([23]). *The hypothesis ETH fails if and only if the parameterized problem MINI(3-SAT) is fixed-parameter tractable.*

The parameterized miniaturization process on SNP-complete problems under the serf-reduction enables the discovery of a class of parameterized problems whose fixed parameterized tractability is equivalent to the failure of the ETH hypothesis, for which the MINI(3-SAT) problem is a typical representative. Based on this observation, Downey et al. [23] introduced a new parameterized class $M[1]$ that consists of all parameterized problems that are fpt-reducible to the MINI(3-SAT) problem. In particular, the fixed parameter tractability of any $M[1]$ -complete problem (under the fpt-reduction) is equivalent to the failure of the ETH hypothesis. The following are some examples of $M[1]$ -complete problems, obtained based on the parameterized miniaturization process on “size-constrained” SNP-complete problems under serf-reductions [30]:

MINI(CIRC-SAT):

Given nonnegative integers k and m in unary, and a circuit C of size bounded by $k \log m$, where k is the parameter, decide if C is satisfiable.

MINI(IS):

Given nonnegative integers k and m in unary, a graph G of size bounded by $k \log m$, and a parameter r , decide if G have an independent set of at least r vertices.

MINI(VC):

Given nonnegative integers k and m in unary, a graph G of size bounded by $k \log m$, and a parameter r , decide if G have a vertex cover of at most r vertices.

The class $M[1]$ has the following relationships with the existing parameterized classes:

Theorem 11 ([23]). $FPT \subseteq M[1] \subseteq W[1]$.

Although the work of Cai and Juedes [9] implicitly hinted at the following result without reference to the class $M[1]$, the result was explicitly stated in Downey et al. [23], and follows from Theorem 10:

Theorem 12 ([23]). $FPT = M[1]$ if and only if 3-SAT is solvable in time $2^{o(n)}$, where n is the number of variables in the input formula, and if and only if the hypothesis ETH fails.

Theorem 12 provides a nice and precise characterization of the subexponential-time computability of many SNP-complete problems (under the self-reduction) in terms of the fixed-parameter tractability of their corresponding miniaturized parameterized problems. However, there are still other important satisfiability problems, whose subexponential-time computability cannot be characterized by the theorem. Observing this, Chen, Flum, and Grohe have further considered the parameterized miniaturization process and derived the equivalence between the subexponential-time computability of problems and the fixed-parameter tractability of their corresponding miniaturized parameterized problems, for higher levels of the W -hierarchy [19–21, 25]. To describe this extension, we have to be further more careful with the notation of the instance size on which the exponential part in the computational complexity of a problem Q is measured. Typically, the *search-size* $\nu(x)$ of an instance x of Q is referred to the cardinality of a universal set U of which the instance x seeks a subset as its solution, and the *length* $|x|$ of x is the length of any reasonable encoding of the instance x . Thus, the computational complexity of the problem Q is measured by a function of the two metrics $\nu(x)$ and $|x|$. Note that in the above case, a simple enumeration algorithm of the subsets of the universal set U solves the problem in time $2^{\nu(x)}|x|^{O(1)}$. We say that the problem Q is *solvable in subexponential-time* if it can be solved in time $2^{o(\nu(x))}|x|^{O(1)}$.

For two integers $t \geq 1$ and $d \geq 1$, let us call a circuit C a $\Pi_{t,d}$ -circuit if C is a Π_{t+1} -circuit in which the fan-in of each gate in level-1 is bounded by the integer constant d . Consider the satisfiability problem $\Pi_{t,d}$ -SAT, which for a given $\Pi_{t,d}$ -circuit C , asks if C is satisfiable. Chen and Grohe [21] introduced the following miniaturized problems:

MINI($\Pi_{t,d}$ -SAT):

Given an instance C of $\Pi_{t,d}$ -SAT, with a parameter $k = \lceil n / \log m \rceil$, where m is the size of C and n is the number of variables in C , decide if C is satisfiable.

We have the following theorem:

Theorem 13 ([21]). *For all $t \geq 1$, the satisfiability problem $\Pi_{t,d}$ -SAT is subexponential-time solvable if and only if the parameterized problem MINI($\Pi_{t,d}$ -SAT) is fixed-parameter tractable.*

Similar to the definition of the class $M[1]$, we can define new miniaturized classes of parameterized problems based on the fpt-reduction:

Definition 1. *For each integer $t \geq 2$, let $M[t]$ be the class of all parameterized problems that are fpt-reducible to the problem MINI($\Pi_{t,d}$ -SAT) for some constant $d \geq 1$.*

Similar to Theorem 11, we obtain:

Theorem 14 ([21]). *For all $t \geq 2$, we have $W[t - 1] \subseteq M[t] \subseteq W[t]$.*

The hierarchy $\bigcup_{t \geq 1} M[t]$ is called the M -hierarchy, which, by Theorem 14, refines the W -hierarchy:

$$FPT \subseteq M[1] \subseteq W[1] \subseteq \dots \subseteq M[t - 1] \subseteq W[t - 1] \subseteq M[t] \subseteq W[t] \dots$$

This study has also motivated the introduction of the following classification in nonparameterized problems that are solvable in exponential time.

Definition 2. *For $t \geq 1$, let $S[t]$ be the class consisting of all the problems that are serf-reducible to the problem $\Pi_{t,d}$ -SAT, for some constant $d \geq 1$.*

For each $t \geq 1$, the parameterized class $M[t]$ is the image of the nonparameterized class $S[t]$ under the miniaturization mapping. The hierarchy $\bigcup_{t \geq 1} S[t]$ is called the S -hierarchy. The miniaturization process serves as a very nice mapping from nonparameterized problems solvable in exponential time to parameterized problems. More specifically, it maps an equivalence class \mathcal{E}_1 of exponential-time solvable problems (under the serf-reduction) to an equivalence class \mathcal{E}_2 of parameterized problems (under the fpt-reduction) such that \mathcal{E}_1 is subexponential-time solvable if and only if \mathcal{E}_2 is fixed parameter tractable. In particular, this mapping induces an “isomorphism” between the S -hierarchy, a nonparameterized complexity class, and the M -hierarchy, a parameterized complexity class [21].

6 Computational Lower Bounds

Theorem 5 shows that if the $W[1]$ -complete problem $wcnf\ 2\text{-SAT}^-$ is solvable in time $f(k)m^{o(k)}$, then ETH fails. Theorem 6 states that for any $t \geq 2$, if the $W[t]$ -complete problem $wcs^*[t]$ is solvable in time $f(k)n^{o(k)}m^{O(1)}$ then the

satisfiability problem $\text{SAT}[t]$ can be solved in subexponential time. Note that the assumptions in these theorems are weaker than that of collapsing the W -hierarchy. On the other hand, they specify more detailed computational time bounds for the problems. Because of the hypothesis ETH, subexponential-time algorithms for 3-SAT and $\text{SAT}[t]$ for all $t \geq 2$ are unlikely. In this sense, Theorems 5 and 6 offer convincing lower bounds on the parameterized complexity for problems that are hard or complete for each level in the W -hierarchy. Interestingly enough, this line of research also implies lower bounds on computational complexity of approximation algorithms for several NP-hard optimization problems. In this section, we shall discuss recent developments in this line of research.

6.1 Lower Bounds on Parameterized Complexity

To discuss lower bounds for parameterized problems, we again need a more careful description of problem instances. For example, the $\text{WCS}^*[t]$ problem now has three different metrics for each of its instance (C, k) : the length m of the instance (C, k) , the parameter k , and the *search-size* n that is the number of input variables in C . We have seen from the previous section that, unless unlikely collapses occur in parameterized complexity theory, the problems $\text{WCS}^*[t]$ require computational time $f(k)n^{\Omega(k)}p(m)$, for any polynomial p and any function f . The dominating term in the time bound depends on the search-size n and the parameter k , instead of the instance length m .

Many well-known NP-hard problems have similar formulations. We list some of them here:

WEIGHTED CNF-SAT (abbreviated **WCNF-SAT**):

Given a CNF formula F , and an integer k , decide if there is an assignment of weight k that satisfies the formula F . Here the search-size is the number of input variables in F .

SET COVER:

Given a collection \mathcal{F} of subsets in a universal set U , and an integer k , decide whether there is a subcollection of k subsets in \mathcal{F} whose union is equal to U . Here the search-size is the cardinality of the collection \mathcal{F} .

HITTING SET:

Given a collection \mathcal{F} of subsets in a universal set U , and an integer k , decide if there is a subset S of k elements in U such that S intersects every subset in \mathcal{F} . Here the search-size is the cardinality of the universal set U .

Many parameterized problems share the property that they seek a subset of k elements in a set of search-size n satisfying certain properties. In most of the problems that we consider, the search space can be easily identified. For example, for the problems **INDEPENDENT SET** and **CLIQUE**, the search space is the vertex set. Thus, each instance of a parameterized problems is associated with a triple (k, n, m) , where k is the parameter, n is the *search-size* of the instance, and m

is the length (of any reasonable encoding) of the instance. We will call such an instance a (k, n, m) -instance.

Theorems 5 and 6 suggest that the $W[1]$ -complete problem $\text{WCNF } 2\text{-SAT}^-$ and the $W[t]$ -complete problem $\text{WCS}^*[t]$ for $t \geq 2$ seem to have very high parameterized complexity. In the following, we introduce a new reduction to identify problems in the corresponding classes that are at least as difficult as these problems.

Definition 3. *A parameterized problem Q is linearly fpt -reducible (shortly fpt_l -reducible) to a parameterized problem Q' if there exist a function f and an algorithm A such that on each (k, n, m) -instance x of Q , the algorithm A produces, in time $f(k)n^{o(k)}m^{O(1)}$, a (k', n', m') -instance x' of Q' , where $k' = O(k)$, $n' = n^{O(1)}$, $m' = m^{O(1)}$, and x is a yes-instance of Q if and only if x' is a yes-instance of Q' .*

The fpt_l -reduction naturally introduces the hardness of parameterized problems.

Definition 4. *A parameterized problem Q_1 is $W[1]$ -hard under the linear fpt -reduction, shortly $W_l[1]$ -hard, if the problem $\text{WCNF } 2\text{-SAT}^-$ is fpt_l -reducible to Q_1 . A parameterized problem Q_t is $W[t]$ -hard under the linear fpt -reduction, shortly $W_l[t]$ -hard, for $t \geq 2$ if the problem $\text{WCS}^*[t]$ is fpt_l -reducible to Q_t .*

Based on the above definitions and using Theorem 5 and Theorem 6, we immediately derive:

Theorem 15. *For $t \geq 2$, no $W_l[t]$ -hard parameterized problem can be solved in time $f(k)n^{o(k)}m^{O(1)}$ for a function f , unless the problem $\text{SAT}[t]$ is solvable in time $2^{o(n)}m^{O(1)}$, which implies the collapsing $W[t - 1] = \text{FPT}$.*

Theorem 16. *No $W_l[1]$ -hard parameterized problem can be solved in time $f(k)m^{o(k)}$ for a function f , unless the ETH hypothesis fails, which is equivalent to the collapsing $M[1] = \text{FPT}$.*

In fact, many known fpt -reductions on parameterized problems proposed in the literature are fpt_l -reductions, or can be modified to become fpt_l -reductions. Using these fpt_l -reductions, we can immediately derive computational lower bounds for a large number of parameterized problems.

Theorem 17. *The following parameterized problems are $W_l[2]$ -hard: WCNF-SAT , SET COVER , HITTING SET , and DOMINATING SET . Thus, unless the problem $\text{SAT}[2]$ is solvable in time $2^{o(n)}m^{O(1)}$, none of them can be solved in time $f(k)n^{o(k)}m^{O(1)}$ for any function f .*

To consider $W_l[1]$ -hard problems, define $\text{WCNF } h\text{-SAT}$, where $h > 0$ is a fixed integer, to be the parameterized problem consisting of the pairs (F, k) , where F is a CNF formula in which each clause is a disjunction of at most h literals and F has a satisfying assignment of weight k .

Theorem 18. *The following problems are $W_1[1]$ -hard: WCNF h -SAT for any integer $h \geq 2$, CLIQUE, and INDEPENDENT SET. Thus, unless the ETH hypothesis fails, none of them can be solved in time $f(k)m^{o(k)}$ for any function f .*

Each of the problems in Theorem 17 and Theorem 18 can be solved by a trivial brute-force algorithm of running time $cn^k m^2$, where c is an absolute constant, which simply enumerates all possible subsets of k elements in the search space. A lot of research has sought new approaches to improve this trivial upper bound. One of the common approaches is to apply a more careful branch-and-bound search process trying to optimize the manipulation of local structures before each branch. Continuously improved algorithms for these problems have been developed based on improved local structure manipulations. It has even been proposed to automate the manipulation of local structures [40] in order to further improve the computational time.

Theorem 17 and Theorem 18, however, provide strong evidence that the power of this approach is quite limited in principle. The lower bound $f(k)n^{\Omega(k)}p(m)$ for the problems in Theorem 17 and the lower bound $f(k)m^{\Omega(k)}$ for the problems in Theorem 18, where f can be any function and p can be any polynomial, indicate that *no* local structure manipulation running in polynomial time or in time depending only on the target value k will obviate the need for exhaustive enumerations.

One might suspect that a particular parameter value (e.g., a very small parameter value or a very large parameter value) would help solving the problems in Theorem 17 and Theorem 18 more efficiently. This possibility is, unfortunately, denied by the following theorems, which indicate that, essentially, the problems are difficult for *every* parameter value.

Theorem 19. *For any constant ϵ , $0 < \epsilon < 1$, and for any nondecreasing and unbounded function μ satisfying $\mu(n) \leq n^\epsilon$, and $\mu(2n) \leq 2\mu(n)$, none of the problems in Theorem 17 can be solved in time $n^{o(k)}m^{O(1)}$ even if we restrict the parameter values k to $\mu(n)/8 \leq k \leq 16\mu(n)$, unless the problem SAT[2] is solvable in time $2^{o(n)}m^{O(1)}$, which implies $W[1] = FPT$.*

Note that the conditions on the function μ in Theorem 19 are satisfied by most complexity functions, such as $\mu(n) = \log \log n$ and $\mu(n) = n^{4/5}$. Therefore, for example, unless the problem SAT[2] is solvable in time $2^{o(n)}m^{O(1)}$, for any polynomial $p(m)$, constructing a hitting set of $\log \log n$ elements requires time $n^{\Omega(\log \log n)}p(m)$, and constructing a hitting set of \sqrt{n} elements requires time $n^{\Omega(\sqrt{n})}p(m)$, where n is the size of the universal set U and m is the instance length.

Similar results hold for the problems in Theorem 18.

Theorem 20. *For any constant ϵ , $0 < \epsilon < 1$, and any nondecreasing and unbounded function μ satisfying $\mu(n) \leq n^\epsilon$, and $\mu(2n) \leq 2\mu(n)$, none of the problems in Theorem 18 can be solved in time $m^{o(k)}$ even if we restrict the parameter values k to $\mu(m)/8 \leq k \leq 16\mu(m)$, unless the ETH hypothesis fails.*

6.2 Refinements and Further Lower Bounds

The lower bounds on parameterized complexity in the previous subsection can be further strengthened based on more careful examinations of the relation between satisfiability problems and parameterized problems. Some of these strengthened results also require a stronger assumption on the complexity of satisfiability problems.

The efforts on achieving faster algorithms for satisfiability have been tremendous [37]. The current best algorithm for the CNF-SAT problem runs in time $2^{n(1-1/O(\log(m/n)))}m^{O(1)}$ [10]. Moreover, the current approaches do not seem to lead to break the time upper bound of the form $2^{n-o(n)}m^{O(1)}$ for solving the problem. In particular, designing an algorithm of running time $2^{\delta n}m^{O(1)}$, where $\delta < 1$ is a constant, seems to require a breakthrough. Impagliazzo and Paturi [30] conjectured that the CNF-SAT problem does not have an algorithm of running time $2^{\delta n}m^{O(1)}$, for a constant $\delta < 1$. Based on this conjecture, stronger computational lower bounds for parameterized problems can be achieved.

First, consider the DOMINATING SET problem: given a graph G of n vertices and a parameter k , decide if the graph G has a dominating set of at most k vertices. It is straightforward to solve the problem in time $O(n^{k+1})$ by enumerating every subset of at most k vertices in the graph and verifying if the subset makes a dominating set for the graph G . Based on fast matrix multiplication algorithms, we can slightly improve the above straightforward enumeration algorithm:

Proposition 1. ([39]) *The DOMINATING SET problem can be solved in time $n^{k+o(1)}$ for $k \geq 7$.*

One may suspect that by applying some algorithmic tricks, we may be able to further improve the algorithm for DOMINATING SET. Note that this was the case for the problem of finding a clique of size k , which can be solved in time $O(n^{(\omega/3)^k}) = O(n^{0.793k})$, where $\omega < 2.376$ is the fastest matrix multiplication exponent. However, Patrascu and Williams have shown that such improvements would lead to a significant advancement in the research on CNF-SAT algorithms:

Theorem 21. ([39]) *For any constant $\epsilon > 0$, the DOMINATING SET problem cannot be solved in time $O(n^{k-\epsilon})$ unless the CNF-SAT problem can be solved in time $2^{\delta n}m^{O(1)}$ for some constant $\delta < 1$.*

Theorem 21 can be extended to other NP-hard problems based on effective reductions. For example, consider the SET COVER problem: given a collection \mathcal{C} of n subsets of a universal set U of size m , decide if there are k subsets in \mathcal{C} whose union is equal to U . Since the DOMINATING SET problem can be easily reduced to the SET COVER problem without changing the parameter value k , we derive directly that the SET COVER problem cannot be solved in time $O(n^{k-\epsilon})$ unless the CNF-SAT problem can be solved in time $2^{\delta n}m^{O(1)}$ for a constant $\delta < 1$.

These techniques have led to computational lower bounds for other interesting problems. The reader is referred to [39] for more results.

Recent research has further considered developing super-linear exponential-time lower bounds on parameterized problems.

A number of well-known parameterized problems went through the process of starting with super-linear exponential-time algorithms before advanced algorithmic techniques were developed that resulted in linear exponential-time algorithms for these problems. For example, the k -PATH problem (given a graph G and a parameter k , decide if the graph contains a simple path of length k) started with an algorithm of running time $2^k k! m^{O(1)}$, proposed in 1985 [35]. It was actually an open problem posted by Papadimitriou and Yannakakis whether k -PATH admits an algorithm running in time $2^{O(k)} m^{O(1)}$. Today, there is number of algorithms, based on at least three different new algorithmic techniques, for the k -PATH problem that run in time $c^k m^{O(1)}$, where c is a small constant [2, 4, 18, 41]. Other examples of this kind include the 3-D MATCHING and the 3-SET PACKING problems [13, 24, 32].

Therefore, the research in exponential-time algorithms could be still in a very premature stage, and one has to be very careful in conjecturing a super-linear exponential-time lower bound for a parameterized problem. On the other hand, very recent research shows that in certain cases, we can derive super-linear exponential-time lower bounds for parameterized problems based on certain beliefs about the complexity of satisfiability problems.

Consider the following CLOSEST STRING problem:

CLOSEST STRING:

Given a set of strings s_1, s_2, \dots, s_t of the same length, and a parameter k , decide if there is a string s of the same length such that the Hamming distance between s and every s_i is bounded by k .

It has been known for a while [28] that the CLOSEST STRING problem can be solved in time $O(2^{k \log k} m)$, where m is the size of the input instance. A natural question is whether the exponential part, i.e., $2^{k \log k} = k^k$, in the complexity $O(2^{k \log k} m)$ can be improved to c^k for a constant c . Recent work by Lokshitanov, Marx, and Saurabh, shows that this is unlikely:

Theorem 22. ([33]) *Unless the hypothesis ETH fails, there is no $2^{o(k \log k)} m^{O(1)}$ -time algorithm for the CLOSEST STRING problem.*

It was shown in [33] that for some other parameters the CLOSEST STRING problem also has super-linear exponential-time lower bounds. Other parameterized problems with super-linear exponential-time lower bounds can also be found in [33].

6.3 Lower Bounds on Approximation Algorithms

An interesting extension of the approach described in the previous subsections is deriving lower bounds on the computational complexity of approximation algorithms for NP-hard problems. We first give a brief review on the terminologies in approximation algorithms.

An NP optimization problem Q is a quadruple (I_Q, S_Q, f_Q, opt_Q) , where

- I_Q is the set of input instances. It is recognizable in polynomial time;
- For each instance $x \in I_Q$, $S_Q(x)$ is the set of feasible solutions for x , which is defined by a polynomial p and a polynomial time computable predicate π (p and π only depend on Q) as $S_Q(x) = \{y : |y| \leq p(|x|) \text{ and } \pi(x, y)\}$;
- $f_Q(x, y)$ is the objective function mapping a pair $x \in I_Q$ and $y \in S_Q(x)$ to a non-negative integer. The function f_Q is computable in polynomial time;
- $opt_Q \in \{\max, \min\}$. Q is called a *maximization problem* if $opt_Q = \max$, and a *minimization problem* if $opt_Q = \min$.

An *optimal solution* y_0 for an instance $x \in I_Q$ is a feasible solution in $S_Q(x)$ such that $f_Q(x, y_0) = opt_Q\{f_Q(x, z) \mid z \in S_Q(x)\}$. We will denote by $opt_Q(x)$ the value $opt_Q\{f_Q(x, z) \mid z \in S_Q(x)\}$.

An algorithm A is an *approximation algorithm* for an NP optimization problem Q if, for each input instance x in I_Q , the algorithm A returns a feasible solution $y_A(x)$ in $S_Q(x)$. The approximation algorithm A has an *approximation ratio* $r(m)$ if for any instance x in I_Q , the solution $y_A(x)$ constructed by the algorithm A satisfies the following condition:

- $opt_Q(x)/f_Q(x, y_A(x)) \leq r(|x|)$ if Q is a maximization problem;
- $f_Q(x, y_A(x))/opt_Q(x) \leq r(|x|)$ if Q is a minimization problem.

An NP optimization problem Q has a *polynomial-time approximation scheme* (PTAS) if there is an algorithm $A_Q(x, \epsilon)$ such that for each fixed real number $\epsilon_0 > 0$, $A_Q(x, \epsilon_0)$ is a polynomial-time approximation algorithm for the problem Q whose approximation ratio is bounded by $1 + \epsilon_0$.

The following “parameterization process” for NP optimization problems has been proposed in the literature.

Definition 5. Let $Q = (I_Q, S_Q, f_Q, opt_Q)$ be an NP optimization problem. The parameterized version of Q is defined as follows:

- If Q is a maximization problem, then the parameterized version of Q is defined as $Q_{\geq} = \{(x, k) \mid x \in I_Q \text{ and } opt_Q(x) \geq k\}$;
- If Q is a minimization problem, then the parameterized version of Q is defined as $Q_{\leq} = \{(x, k) \mid x \in I_Q \text{ and } opt_Q(x) \leq k\}$.

The above definition offers the possibility to study the relationship between the approximability and the parameterized complexity of NP optimization problems.

Theorem 23. Let Q be an NP optimization problem. If the parameterized version of Q is $W_1[1]$ -hard, then Q has no PTAS of running time $f(1/\epsilon)m^{o(1/\epsilon)}$ for any function f , unless the ETH hypothesis fails.

Proof. We consider the case that $Q = (I_Q, S_Q, f_Q, opt_Q)$ is a maximization problem such that the parameterized version Q_{\geq} of Q is $W_1[1]$ -hard.

Suppose to the contrary that Q has a PTAS A_Q of running time $f(1/\epsilon)m^{o(1/\epsilon)}$ for a function f . We show how to use the algorithm A_Q to solve the parameterized problem Q_{\geq} . Consider the following algorithm A_{\geq} for Q_{\geq} :

Algorithm A_{\geq} :

On an instance (x, k) of Q_{\geq} , call the PTAS algorithm A_Q on the instance x of Q with the real number $\epsilon = 1/(2k)$. Suppose that A_Q returns a solution y in $S_Q(x)$. If $f_Q(x, y) \geq k$, then return “yes”, otherwise return “no”.

We verify that the algorithm A_{\geq} solves the parameterized problem Q_{\geq} . Since Q is a maximization problem, if $f_Q(x, y) \geq k$ then obviously $opt_Q(x) \geq k$. Thus, the algorithm A_{\geq} returns a correct decision in this case. On the other hand, suppose $f_Q(x, y) < k$. Since $f_Q(x, y)$ is an integer, we have $f_Q(x, y) \leq k - 1$. Since A_Q is a PTAS for Q and $\epsilon = 1/(2k)$, we must have

$$opt_Q(x)/f_Q(x, y) \leq 1 + 1/(2k).$$

From this we get (note that $f_Q(x, y) < k$)

$$opt_Q(x) \leq f_Q(x, y) + f_Q(x, y)/(2k) \leq k - 1 + 1/2 = k - 1/2 < k.$$

Thus, in this case the algorithm A_{\geq} also returns a correct decision. This proves that the algorithm A_{\geq} solves the parameterized version Q_{\geq} of the problem Q . The running time of the algorithm A_{\geq} is dominated by that of the algorithm A_Q , which by our hypothesis is bounded by $f(1/\epsilon)m^{\sigma(1/\epsilon)} = f(2k)m^{\sigma(k)}$. Thus, the $W_1[1]$ -hard problem Q_{\geq} is solvable in time $f(2k)m^{\sigma(k)}$. By Theorem 16, this implies that the ETH hypothesis fails.

The proof is similar for the case when Q is a minimization problem, and hence is omitted.

We demonstrate an application for Theorem 23. We pick the problem DISTINGUISHING SUBSTRING SELECTION as an example, which has drawn a lot of attention recently because of its applications in computational biology such as in drug generic design [22].

Consider all strings over a fixed alphabet. Denote by $|s|$ the length of the string s . The *distance* $D(s_1, s_2)$ between two strings s_1 and s_2 , $|s_1| \leq |s_2|$, is defined as follows. If $|s_1| = |s_2|$, then $D(s_1, s_2)$ is the Hamming distance between s_1 and s_2 ; and if $|s_1| < |s_2|$, then $D(s_1, s_2)$ is the minimum of $D(s_1, s'_2)$ over all substrings s'_2 of length $|s_1|$ in s_2 .

Based on the standard formulation of NP optimization problems, the (optimization version of the) DISTINGUISHING SUBSTRING SELECTION problem (DSSP) is defined as follows:

Definition 6. The DSSP problem is a quadruple (I_D, S_D, f_D, opt_D) , where

- The instance set I_D is the set of tuples of the form (n, S_b, S_g, d_b, d_g) , where n , d_b , and d_g are integers, $d_b \leq d_g$, $S_b = \{b_1, \dots, b_{n_b}\}$ is a set of (bad) strings, $|b_i| \geq n$, and $S_g = \{g_1, \dots, g_{n_g}\}$ is a set of (good) strings, $|g_j| = n$;
- For an instance $x = (n, S_b, S_g, d_b, d_g)$ in I_D , the solution set $S_D(x)$ consists of all strings of length n ;

- For an instance $x = (n, S_b, S_g, d_b, d_g)$ in I_D and a solution $s \in S_D(x)$, the objective function value $f_D(x, s)$ is defined to be the largest non-negative integer d such that (i) $d \leq d_g$; (ii) $D(s, b_i) \leq d_b(2 - d/d_g)$ for all $b_i \in S_b$; and (iii) $D(s, g_j) \geq d$ for all $g_j \in S_g$. If such an integer d does not exist, then define $f_D(x, s) = 0$;
- $opt_D = \max$

Note that for $x \in I_D$ and $s \in S_D(x)$, the value $f_D(x, s)$ can be computed in polynomial time by checking each number $d = 0, 1, \dots, d_g \leq n$.

Note that the objective of the DSSP problem is to find a string s that maximizes the value $f_D(x, s)$, which is bounded by d_g . In particular, if a string s can achieve $f_D(x, s) = d_g$, then s satisfies $D(s, b_i) \leq d_b$ for all $b_i \in S_b$ (i.e., the string s is similar enough to all bad strings) and $D(s, g_j) \geq d_g$ for all $g_j \in S_g$ (i.e., the string s is sufficiently different from all good strings).

The DSSP problem is NP-hard [27]. Deng et al. [22] developed a PTAS for DSSP whose running time is bounded by $O(m(n_b + n_g)^{O(1/\epsilon^6)})$, where m is the size of the instance.³ Obviously, such an algorithm is not practical even for moderate values of the error bound ϵ . The question is, can we develop significantly faster PTAS for the DSSP problem?

Using the above parameterization process, we can parameterize the DSSP problem, and study the complexity of the corresponding parameterized problem $DSSP_{\geq}$.

Lemma 3. ([14]) *The parameterized problem $DSSP_{\geq}$ is $W_1[1]$ -hard.*

From Lemma 3 and Theorem 23, we get the following result.

Theorem 24. *Unless the ETH hypothesis fails, the problem $DSSP$ has no PTAS whose running time is bounded by $f(1/\epsilon)m^{o(1/\epsilon)}$ for any function f .*

Therefore, Theorem 24 implies that any PTAS for DSSP cannot run in time $f(1/\epsilon)m^{o(1/\epsilon)}$ for any function f . Thus essentially, no PTAS for DSSP can be practically efficient even for moderate values of the error bound ϵ . This is the first time a specific lower bound is derived on the running time of a PTAS for an NP-hard problem.

Lemma 3 is proved by a linear fpt-reduction from the DOMINATING SET problem to the problem $DSSP_{\geq}$, which leads to the computational lower bounds on PTAS for the DSSP problem in Theorem 24. This approach demonstrates an interesting property of this technique. In most cases, computational lower bounds and inapproximability of optimization problems are derived based on approximation ratio-preserving reductions [3], by which if a problem Q_1 is reduced to another problem Q_2 , then Q_2 is at least as hard as Q_1 . In particular, if

³ In fact, the formulations of the optimization versions of the DSSP problem and its PTAS given in [22] look very different from the versions presented here. A proof is given in [14] that shows the equivalences of the problem formulation and PTAS given in [22] and that presented here.

Q_1 is reduced to Q_2 under an approximation ratio-preserving reduction, then the approximability of Q_2 is at least as difficult as that of Q_1 . Therefore, the intractability of an “easier” problem in general cannot be derived using such a reduction from a “harder” problem. On the other hand, our computational lower bound on DSSP was obtained by a linear fpt-reduction from DOMINATING SET. It is well-known that DOMINATING SET has no polynomial time approximation algorithms of constant ratio [3], while DSSP has a PTAS. Thus, from the viewpoint of approximation, DOMINATING SET is much harder than DSSP, and our linear fpt-reduction reduces a harder problem to an easier problem. This hints that this approach for deriving computational lower bounds *cannot* be simply replaced by the standard approaches based on approximation ratio-preserving reductions.

Readers who are interested in the relation between fixed-parameter tractability and the efficiency of approximation algorithms for NP-hard optimization problems are referred to [8, 15, 34] for more discussions and details.

7 Concluding Remarks and Open Problems

The study of parameterized intractability and subexponential-time computability has significantly promoted new research directions in both complexity theory and algorithms. From the computational complexity viewpoint, this study has motivated the development of new frameworks whose intrinsic relations are being studied. From the algorithmics viewpoint, this study has motivated the invention of new algorithmic tools, beyond the world of polynomial-time computation, and has established connections among a large variety of computational problems. This demonstrates the robustness of computational intractability, thus providing convincing evidence of the existing computational lower bounds on problems of theoretical and practical importance, based on parameterized complexity and subexponential-time hypotheses.

The classification of parameterized intractability, i.e., the W -hierarchy, offers a framework for the study of computational intractability, which is a refinement of that of classical complexity theory. On the other hand, little is known about the structural properties of this hierarchy. For example, any natural “hierarchy collapsing” results about the W -hierarchy are still lacking. In particular, the following question remains unanswered: Does $FPT = W[t]$ ($t \geq 1$) imply $FPT = W[s]$ for $s > t$? Note that because of the close connections between the W -hierarchy and the computational complexity of circuit satisfiability, the corresponding questions about circuit satisfiability problems are also important and significant: Would the subexponential-time computability of $SAT[t]$ imply the subexponential-time computability of $SAT[s]$ for $s > t$? In particular, would the failure of the ETH hypothesis imply the subexponential-time computability of CNF-SAT?

The current state of knowledge about the complexity of satisfiability problems seems to provide no hints on the above questions. For example, the 3-SAT problem can be solved in time $2^{\delta n} m^{O(1)}$, where $\delta < 0.56$ [5]. However, this does

not seem to offer any ideas for solving the general CNF-SAT problem in time $2^{\delta'n}m^{O(1)}$ for any constant $\delta' < 1$. In fact, the current techniques used for solving 3-SAT do not even seem to be generalizable to CNF-SAT with bounded fan-in of level-1 gates (i.e., h -SAT for integer constants $h \geq 3$). It will be very interesting to investigate this direction. For example, is there a relation between the ETH hypothesis and the hypothesis “CNF-SAT is not solvable in time $2^{\delta'n}m^{O(1)}$ for any constant $\delta < 1$ ”? Would a subexponential-time algorithm for 3-SAT imply a $2^{\delta'n}m^{O(1)}$ -time algorithm for CNF-SAT for some constant $\delta < 1$, or vice versa?

This reminds us of the well-known research line in complexity theory on the computational lower bounds for bounded-depth circuit computation, where it has been known that depth- t circuits are strictly more powerful than depth- s circuits for $t > s$ [29]. In fact, the fan-in of level-1 gates of a circuit is determinative of its computational power: for any $h \geq 2$, depth- h circuits in which the fan-in of level-1 gates is bounded by t is strictly more powerful than depth- h circuits in which the fan-in of level-1 gates is bounded by s , if $t > s$ [7]. Note that, these significant results are on the difference of computational power of circuit models. On the other hand, in the study of parameterized complexity and the complexity of satisfiability problems, the circuit depths in the problems are part of the “descriptive complexity” of the problems. Are there any correlations that can provide further insight in this direction? If we view a nondeterministic computation as a “guess-then-check” process [6], then the SAT[t] problems for all $t \geq 2$ require the same guessing power (i.e., picking a proper subset of the input variables), but differ strictly in the verification power (SAT[t] requires depth- t circuits for verification). The current research status in satisfiability algorithms seems to suggest that the difference in the verification power forces a difference in the deterministic computational complexity of the problems.

Another interesting research direction is on the algorithmics side. As stated in Theorem 21, a very sharp parameterized lower bound (such as $\Omega(n^k)$) for the DOMINATING SET problem seems to have consequences on exact algorithms for the important satisfiability problem CNF-SAT. This line of research has yielded a collection of results, formulated as “if problem A can be solved in time $t(\cdot)$ then problem B can be solved in time $s(\cdot)$,” where $t(\cdot)$ and $s(\cdot)$ are precise functions (without hidden constants in their asymptotic notations). Is there a systematic method to relate such results? Such method will bear significant impact on the existence of more efficient exact algorithms for certain problems, and can be read from two angles. From the positive angle, those results suggest a way for improving the algorithms for a problem B under a certain framework (such as exact computability) by improving the algorithms for another problem A with respect to (possibly) a different computational framework (such as parameterized complexity). From the negative angle, such a result can serve as an indicator of the intricate difficulty of the computability of a problem A with respect to a certain computational framework, based on that of another problem B with respect to a different framework.

We also want to remark that the above discussions provide “observations” based on the current understanding and techniques in complexity theory and

algorithmic research. For instance, the statement that the CNF-SAT problem cannot be solved in time $2^{\delta n} m^{O(1)}$ for a constant $\delta < 1$ seems to be very strong, and perhaps needs further investigation. One has to be more careful when using such statements as “hypotheses.” Instead, these studies and observations should provide an impetus for new research insights, new ideas, and new techniques. All our algorithmic techniques for solving NP-hard problems, such as the satisfiability problems, are more or less based on enumerations. Thus, a more ambitious question would be to investigate new approaches for tackling NP-hard problems.

References

1. Abrahamson, K., Downey, R., Fellows, M.: Fixed-parameter tractability and completeness IV: On completeness for W[P] and PSPACE analogues. *Annals of Pure and Applied Logic* 73(3), 235–276 (1995)
2. Alon, N., Yuster, R., Zwick, U.: Color-coding. *Journal of the ACM* 42, 844–856 (1995)
3. Ausiello, G., Protasi, M., Marchetti-Spaccamela, A., Gambosi, G., Crescenzi, P., Kann, V.: *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties*, 1st edn. Springer-Verlag New York, Inc., Secaucus (1999)
4. Björklund, A.: Determinant sums for undirected hamiltonicity. In: *Proceedings of the 51th Annual IEEE Symposium on Foundations of Computer Science*, pp. 173–182 (2010)
5. Brüggemann, T., Kern, W.: An improved deterministic local search algorithm for 3SAT. *Theoretical Computer Science* 329, 303–313 (2004)
6. Cai, L., Chen, J.: On the amount of nondeterminism and the power of verifying. *SIAM Journal on Computing* 26(3), 733–750 (1997)
7. Cai, L., Chen, J., Håstad, J.: Circuit bottom fan-in and computational power. *SIAM Journal on Computing* 27(2), 341–355 (1998)
8. Cai, L., Fellows, M., Juedes, D., Rosamond, F.: The complexity of polynomial-time approximation. *Theory of Computing Systems* 41(3), 459–477 (2007)
9. Cai, L., Juedes, D.: On the existence of subexponential parameterized algorithms. *Journal of Computer and System Sciences* 67(4), 789–807 (2003)
10. Calabro, C., Impagliazzo, R., Paturi, R.: A duality between clause width and clause density for SAT. In: *IEEE Conference on Computational Complexity*, pp. 252–260 (2006)
11. Chen, J.: Characterizing parallel hierarchies by reducibilities. *Information Processing Letters* 39(6), 303–307 (1991)
12. Chen, J., Chor, B., Fellows, M., Huang, X., Juedes, D., Kanj, I., Xia, G.: Tight lower bounds for certain parameterized NP-hard problems. *Information & Computation* 201(2), 216–231 (2005)
13. Chen, J., Feng, Q., Liu, Y., Lu, S., Wang, J.: Improved deterministic algorithms for weighted matching and packing problems. *Theoretical Computer Science* 412(23), 2503–2512 (2011)
14. Chen, J., Huang, X., Kanj, I., Xia, G.: Strong computational lower bounds via parameterized complexity. *Journal of Computer and System Sciences* 72(8), 1346–1367 (2006)
15. Chen, J., Huang, X., Kanj, I., Xia, G.: Polynomial time approximation schemes and parameterized complexity. *Discrete Applied Mathematics* 155(2), 180–193 (2007)

16. Chen, J., Kanj, I., Jia, W.: Vertex cover: further observations and further improvements. *Journal of Algorithms* 41, 280–301 (2001)
17. Chen, J., Kanj, I., Xia, G.: On parameterized exponential time complexity. *Theoretical Computer Science* 410(27-29), 2641–2648 (2009)
18. Chen, J., Kneis, J., Lu, S., Molle, D., Richter, S., Rossmanith, P., Sze, S.-H., Zhang, F.: Randomized divide-and-conquer: improved path, matching, and packing algorithms. *SIAM Journal on Computing* 38, 2526–2547 (2009)
19. Chen, Y., Flum, J.: On miniaturized problems in parameterized complexity theory. *Theoretical Computer Science* 351(3), 314–336 (2006)
20. Chen, Y., Flum, J.: Subexponential time and fixed-parameter tractability: Exploiting the miniaturization mapping. *Journal of Logic and Computation* 19(1), 89–122 (2009)
21. Chen, Y., Grohe, M.: An isomorphism between subexponential and parameterized complexity theory. *SIAM Journal on Computing* 37(4), 1228–1258 (2007)
22. Deng, X., Li, G., Li, Z., Ma, B., Wang, L.: Genetic design of drugs without side-effects. *SIAM Journal on Computing* 32, 1073–1090 (2003)
23. Downey, R., Estivill-Castro, V., Fellows, M., Prieto-Rodriguez, E., Rosamond, F.: Cutting up is hard to do: the parameterized complexity of k -Cut and related problems. *Electronic Notes in Theoretical Computer Science* 78, 205–218 (2003)
24. Downey, R., Fellows, M.: *Parameterized Complexity*. Springer, New York (1999)
25. Flum, J., Grohe, M.: Parameterized complexity and subexponential time (column: Computational complexity). *Bulletin of the EATCS* 84, 71–100 (2004)
26. Flüm, J., Grohe, M.: *Parameterized Complexity Theory*. Springer, Berlin (2010)
27. Gramm, J., Guo, J., Niedermeier, R.: On Exact and Approximation Algorithms for Distinguishing Substring Selection. In: Lingas, A., Nilsson, B.J. (eds.) *FCT 2003*. LNCS, vol. 2751, pp. 195–209. Springer, Heidelberg (2003)
28. Gramm, J., Niedermeier, R., Rossmanith, P.: Fixed-parameter algorithms for closest string and related problems. *Algorithmica* 37(1), 25–42 (2003)
29. Håstad, J.: Computational limitations of small depth circuits. Technical report, MIT Press (1986)
30. Impagliazzo, R., Paturi, R.: On the complexity of k -SAT. *Journal of Computer and System Sciences* 62(2), 367–375 (2001)
31. Impagliazzo, R., Paturi, R., Zane, F.: Which problems have strongly exponential complexity? *Journal of Computer and System Sciences* 63(4), 512–530 (2001)
32. Koutis, I.: Faster Algebraic Algorithms for Path and Packing Problems. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfssdóttir, A., Walukiewicz, I. (eds.) *ICALP 2008, Part I*. LNCS, vol. 5125, pp. 575–586. Springer, Heidelberg (2008)
33. Lokshtanov, D., Marx, D., Saurabh, S.: Slightly superexponential parameterized problems. In: *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 760–776 (2011)
34. Marx, D.: Parameterized complexity and approximation algorithms. *The Computer Journal* 51(1), 60–78 (2008)
35. Monien, B.: How to find long paths efficiently. *Ann. Discrete Math.* 25, 239–254 (1985)
36. Niedermeier, R.: *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, USA (2006)
37. International Conference on Theory and Applications of Satisfiability Testing, <http://www.satisfiability.org/>
38. Papadimitriou, C., Yannakakis, M.: Optimization, approximation, and complexity classes. *Journal of Computer and System Sciences* 43, 425–440 (1991)

39. Patrascu, M., Williams, R.: On the possibility of faster SAT algorithms. In: Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 1065–1075 (2010)
40. Robson, J.M.: Algorithms for maximum independent sets. *Journal of Algorithms* 7, 425–440 (1986)
41. Williams, R.: Finding paths of length k in $O^*(2^k)$ time. *Information Processing Letters* 109(6), 315–318 (2009)
42. Williams, R.: Non-uniform ACC circuit lower bounds. In: IEEE Conference on Computational Complexity, pp. 115–125. IEEE Computer Society Press (2011)
43. Woeginger, G.: Exact Algorithms for NP-Hard Problems: A Survey. In: Jünger, M., Reinelt, G., Rinaldi, G. (eds.) *Combinatorial Optimization (Edmonds Festschrift)*. LNCS, vol. 2570, pp. 185–207. Springer, Heidelberg (2003)

Fixed-Parameter Tractability of Treewidth and Pathwidth

Hans L. Bodlaender

Department of Information and Computing Sciences, Utrecht University,
P.O. Box 80.089, 3508 TB Utrecht, The Netherlands
`h.l.bodlaender@uu.nl`

Abstract. In this survey, a number of results on the fixed-parameter tractability of treewidth and pathwidth are discussed. Some emphasis is placed on older results, and proofs that show that treewidth and pathwidth are fixed-parameter tractable. Also, a linear-time algorithm for testing if a graph has pathwidth at most some given constant is discussed in more detail.

1 Introduction

This overview paper is on the occasion of the 60th birthday of Mike Fellows. Already in the early development of the theory reported here, Mike's insights were at many points of great importance, and his work and his enthusiasm for the topics were always a great source of inspiration. Many of the ideas discussed in this survey were obtained from or inspired by discussions with or talks by Mike Fellows.

Treewidth, and related notions, like pathwidth, branchwidth, cliquewidth, rankwidth play an important role in many modern investigations in algorithmic graph theory, and already from its early origins, in the field of parameterized algorithms. In this survey, a look will be taken at the results that show that the problems to decide if the treewidth or pathwidth of a given graph is at most a given number k are fixed-parameter tractable. This question is an interesting one, for several reasons: the result is used as a subroutine in many recent results, and the investigations for these notions show many important techniques from the field of parameterized algorithms, and often the problem was one of the sources of inspiration for inventing these techniques.

The notions of treewidth and pathwidth were introduced by Robertson and Seymour [110, 113] in their fundamental work on graph minors. However, other, equivalent notions were invented independently, and sometimes earlier by many different authors. Already in the 1960's, it was observed that many problems that are intractable on general graphs become easier to solve on trees and series-parallel graph. Several authors independently noted that these results can be generalized to larger classes of graphs. E.g., Wimer introduced in the 1980's the notion of k -terminal recursive graph classes [143]. Trees can be formed by 'gluing' 1-terminal graphs together; series-parallel graphs by 'gluing' 2-terminal graphs

together; and a similar algorithmic behavior is obtained when using some other constant number of vertices. An often used equivalent version of treewidth is the notion of *partial k -trees* by Arnborg et al. [3, 7]. An overview of several notions that are equivalent (or imply a constant bound) to treewidth or pathwidth can be found in [16].

Nowadays, the notion of treewidth plays a role in many different fields of algorithms research and graph theory. One important reason for the interest is that many problems that are intractable (e.g., NP-hard) become linear time (or sometimes polynomial time) solvable when restricted to graphs of bounded treewidth. Such algorithms have been found for many combinatorial problems (see e.g., [8, 9, 33, 94, 138, 144]), and also have been employed for problems from computational biology (see e.g., [100]), constraint satisfaction (see e.g., [40, 47, 78, 94]), and probabilistic networks (see [99]). See e.g., also [3, 2, 6, 12, 32, 39, 38, 42, 50, 70, 79, 80, 82, 85, 88, 106, 105, 108, 145]. In other words: many graph problems become fixed-parameter tractable when parameterized by the treewidth of the input graph.

This survey is further organized as follows. Section 2 gives some definitions, discusses equivalent notions, and some well known lemmas on treewidth and pathwidth. In Section 3, linear-time algorithms for problems on graphs of bounded treewidth are discussed, including the 'automata view' on these algorithms, pioneered by Fellows and Langston. Section 4 discusses some algorithmic consequences of the theory of graph minors. Section 5 looks at the complexity of deciding treewidth and pathwidth, with most emphasis on the fixed-parameter case. It includes a pathwidth version of the result that the fixed-parameter case of treewidth is linear time solvable ([14]). The paper ends with short conclusions and a few major open problems.

For easier presentation, some arguments and proofs have technical inaccuracies, and at some points, an overload of notation seemed unavoidable. I still hope that many of the ideas and techniques come across.

2 Definitions and Equivalent Notions

Throughout this paper, n denotes the number of vertices of graph $G = (V, E)$. Unless otherwise stated, graphs are considered to be simple and undirected. Several of the results generalize to directed graphs, but this will not be elaborated here.

The notions of treewidth and tree decomposition were introduced by Robertson and Seymour [113] in their fundamental work on graph minors.

Definition 1. *A tree decomposition of a graph $G = (V, E)$ is a pair $(\{X_i \mid i \in I\}, T = (I, F))$, with $\{X_i \mid i \in I\}$ a family of subsets of V (called bags) and T a tree, such that*

- $\bigcup_{i \in I} X_i = V$,
- for all $\{v, w\} \in E$, there is an $i \in I$ with $v, w \in X_i$, and
- for all $v \in V$, the set $I_v = \{i \in I \mid v \in X_i\}$ forms a connected subtree of T .

The width of tree decomposition $(\{X_i \mid i \in I\}, T = (I, F))$ is $\max_{i \in I} |X_i| - 1$. The treewidth of a graph G , $tw(G)$, is the minimum width among all tree decompositions of G .

The third condition in the definition above can be replaced by the following equivalent condition:

For all $i_1, i_2, i_3 \in I$: if i_2 is on the path from i_1 to i_3 in T , then $X_{i_1} \cap X_{i_3} \subseteq X_{i_2}$.

An example of a graph with a tree decomposition can be found in Figure 1.

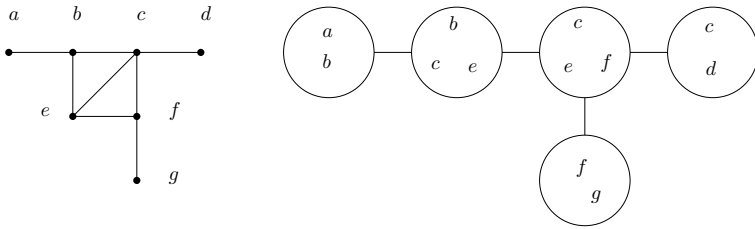


Fig. 1. A graph with a tree decomposition

A *path decomposition* is a tree decomposition $(\{X_i \mid i \in I\}, T = (I, F))$ with T a path. The pathwidth of a graph is the minimum width of a path decomposition of G .

There are several equivalent characterizations of the notions of treewidth and pathwidth. For an overview, see e.g. [16]. We mention a few below that are useful for the further exposition of technical ideas.

2.1 Nice Tree and Path Decompositions

A tree decomposition $(\{X_i \mid i \in I\}, T = (I, F))$ is *nice*, if T is a rooted tree, and each node is of one of the four following types:

- **Leaf:** a leaf node i has no children and has $|X_i| = 1$, i.e., a bag size of one.
- **Join:** a join node i has two children j_1, j_2 with $X_i = X_{j_1} = X_{j_2}$, i.e., with the same bags.
- **Forget:** a forget node i has one child j such that there is a $v \in V$ with $X_i = X_j - \{v\}$.
- **Introduce:** a **forget** node i has one child j such that there is a $v \in V$ with $X_i = X_j \cup \{v\}$.

It is well known that one can transform a tree decomposition into a nice one with the same width, and with $O(n)$ nodes, in linear time. One of the first occurrences of nice tree decompositions is in [91].

We similarly have *nice path decompositions*. A nice path decomposition can be represented by a series of bags (X_0, \dots, X_r) with $r = O(n)$, and with $X_0 = \emptyset$ (the only **leaf** node) and each $X_i, i > 1$ has a vertex v with $X_i = X_{i-1} - \{v\}$ (**forget** nodes) or $X_i = X_{i-1} \cup \{v\}$ (**introduce nodes**). While X_0 is not necessary, using an empty first bag helps for easier notation later on. One can show the following result.

Theorem 1. *Suppose we are given a graph $G = (V, E)$ and a tree (path) decomposition of $G, (\{X_i \mid i \in I\}, T = (I, F))$ of G of width k . Then a nice tree (path) decomposition of G of width k can be found in $O(k(n + |I|))$ time, such that the nice tree decomposition has $O(n)$ bags.*

2.2 k -Terminal Graphs

A *terminal graph* is a triple (V, E, X) with $X \subseteq V$ an ordered set of vertices, called the *terminals*. (V, E, X) is a k -terminal graph if $|X| = k$. We define the \oplus operation on pairs of k -terminal graphs: $(V, E, X) \oplus (W, F, Y)$ is obtained by taking the disjoint union of (V, E) and (W, F) and then identifying the i th terminal of X with the i th terminal of Y ; dropping parallel edges if existing.

For the description of algorithms, it is useful to associate terminal graphs (forming subgraphs of G) with nodes in a nice tree or path decomposition, in the following way. Consider a nice tree decomposition $(\{X_i \mid i \in I\}, T = (I, F))$. For each node $i \in I$, we associate a terminal graph $G_i = (V_i, E_i, X_i)$, with V_i the union of all bags X_j with $j = i$ or j a descendant of i ; and $E_i = \{\{v, w\} \in E \mid v, w \in V_i\}$, (taking some arbitrary ordering on X_i).

For a **leaf** node, the graph G_i simply consists of the unique vertex in X_i and no edges.

If i is a **join** node with children j_1 and j_2 , then the graph G_i can be obtained from the graphs G_{j_1} and G_{j_2} by taking the disjoint union and then identifying the vertices in $X_i = X_{j_1} = X_{j_2}$, i.e., $G_i = G_{j_1} \oplus G_{j_2}$. An example is given in Figure 2.

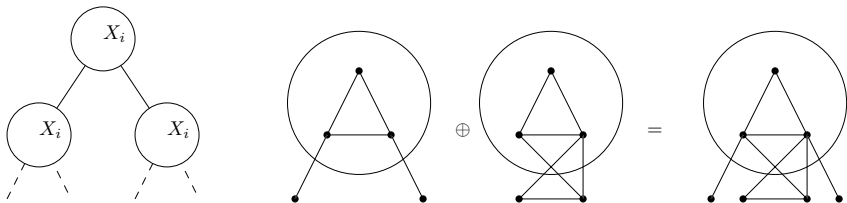


Fig. 2. The \oplus operation, a **join** node and the corresponding subgraphs

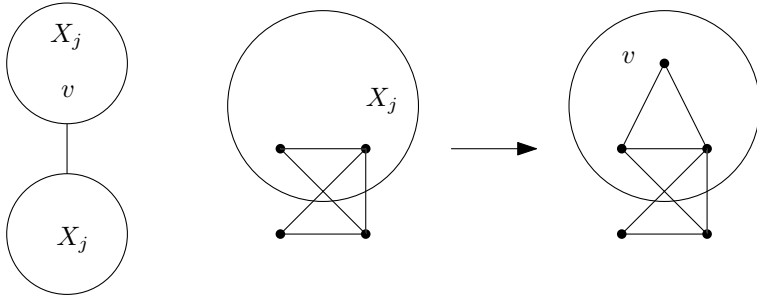


Fig. 3. An **introduce** node and the corresponding subgraphs

If i is an **introduce** node with child j with $X_i = X_j \cup \{v\}$, then G_i is formed from G_j by adding the vertex v and some edges between v and vertices in X_j . See for an example Figure 3.

For a **forget** node, the situation is simple. If i is a **forget** node with child j and $X_i = X_j - \{v\}$, then G_i and G_j have the same vertices and edges; the only difference is that v is no longer a terminal.

2.3 Representing Nice Path Decompositions by Strings

Suppose we consider graphs up to isomorphism, i.e., we ignore vertex names. Then a nice path decomposition of a graph can be represented by a finite string of characters. Assume the vertices in a bag are ordered by the order in which they were introduced. An **introduce** node can now be characterized by the subset of the indices of the neighbors of the introduced vertex. E.g., if $X_i = \{v, w, x\}$ and $X_{i+1} = \{v, w, x, y\}$, with v introduced in a bag with a smaller index than the bag where w is introduced, and w is likewise before x , then if $\{v, y\}$ and $\{x, y\}$ are edges, and $\{w, y\}$ is not an edge in G , then we can characterize node $i + 1$ by the subset $\{1, 3\}$. **Forget** nodes can be characterized by the index of the forgotten vertex. E.g., if in our example, $X_{i+2} = \{v, w, y\}$, then node $i + 2$ can be characterized by the index 3. In this way, we can characterize a nice path decomposition of width at most k by a sequence of at most $2n$ subsets of $\{1, 2, \dots, k\}$ and elements from $\{1, \dots, k + 1\}$, i.e., by a string of length at most $2n$ from an alphabet A_k of size $2^k + k + 1$. This representation has some important consequences: several algorithms that exploit (nice) path decompositions can be represented as finite state automata. More on this in Section 3.2.

Similarly, (nice) tree decompositions can be represented as a labeled tree, and several algorithms on (nice) tree decomposition can be represented as a *finite state tree automaton*. These latter are generalizations of finite state automata, but have as input a labeled tree instead of a string. This view was pioneered by Fellows and Langston [65]. Recently, the approach was moved to the notion of rankwidth by Ganian and Hliněný [76].

Not every string in A_k^* represents a graph of pathwidth at most k . A symbol that tells that we forget the i th terminal should only occur when there are at least i terminals; a symbol that tells that we introduce a vertex with edges to terminals with indices in $I \subseteq \{1, \dots, k\}$ should only occur when we have at most k terminals, and each index in I corresponds to an existing terminal. It is a trivial exercise to see that the set of strings that correspond to a graph of pathwidth at most k is regular, i.e., that we can build a finite state automaton that recognizes this set. Similarly, labeled trees that correspond to tree decompositions of width k can be recognized by a finite state tree automaton.

2.4 Notions Equivalent to Pathwidth

Two other notions that are equivalent with pathwidth are the following. A linear ordering of a graph $G = (V, E)$ is a bijective function $f : V \rightarrow \{1, \dots, n\}$.

Definition 2. *The vertex separation number of a linear ordering f of a graph G equals*

$$\max_{v \in V} |\{w \in V \mid f(w) \leq f(v) \wedge \exists \{w, x\} \in E : f(x) > f(v)\}|$$

The vertex separation number of a graph G equals the minimum vertex separation number of a linear ordering of G .

Theorem 2 (Kinnersley [89]). *The vertex separation number of a graph G equals the pathwidth of G .*

We also have the following folklore result. For a proof, see e.g. [16].

Theorem 3. *Let G be a graph. The pathwidth of G is at most k , if and only if $G = (V, E)$ is a subgraph of an interval graph $H = (V, F)$ with maximum clique size at most $k + 1$.*

2.5 Minors

Another important notion for the theory of treewidth is the notion of *minor*, see e.g., [110]. A graph H is a minor of a graph G , if H can be obtained from G by zero or more of the following operations: removing vertices, removing edges, and contracting edges (an edge contraction replaces two adjacent vertices by one that is incident to the neighbors of the contracted vertices).

More on graph minors in Section 4.

2.6 Cliques

A folklore result on treewidth is often of great help. As observed in [28], it directly follows from the Helly property for trees.

Lemma 1. *Let $(\{X_i \mid i \in I\}, T = (I, F))$ be a tree decomposition of $G = (V, E)$ and let $W \subseteq V$ be a clique in G . Then there exists an $i \in I$ with $W \subseteq X_i$.*

3 Algorithms on Tree and Path Decompositions

One of the most important reasons for the interest in the notion of treewidth (or its related notions) is that many problems become polynomial, and often linear solvable on graphs with some constant upper bound on their treewidth. See e.g., [8, 9, 94, 138, 144].

Most of these algorithms employ dynamic programming in some form. These algorithms consist of two steps. In the first step, a tree decomposition of bounded width is found. This step will be discussed in more detail in Section 5. The tree decomposition then is transformed to a nice one with the same width, with a linear number of nodes, cf. Section 2. In the second step, the (nice) tree decomposition is exploited: in some bottom-up order (e.g., postorder), a table is computed for each node of the tree. To compute a table for a node, all what is (usually) needed is the information of the nodes of its children and a little "local" information (e.g., what vertices in the bag of the node are incident). The problem then can be decided using the table of the root. Construction versions often can be solved by going top-down in the tree, using the information stored in the tables.

Our example of the algorithm uses the 3-COLORING problem.. A 3-coloring of a graph $G = (V, E)$ is a function $c : V \rightarrow \{1, 2, 3\}$ such that for all $\{v, w\} \in E$, $c(v) \neq c(w)$. In the 3-COLORING problem, we are given a graph $G = (V, E)$, and have to decide if there exists a 3-coloring of G .

3.1 Solving 3-COLORING on Nice Tree Decompositions

For the 3-COLORING problem, we compute for each node in the tree decomposition $i \in I$, a table A_i . The table has an entry for each function $f : X_i \rightarrow \{1, 2, 3\}$. The entry maps to a Boolean value, and expresses if the function f can be extended to a 3-coloring of G_i . I.e., $A_i(f)$ is true, if and only if there exists a 3-coloring g of G_i such that for all $v \in X_i$, $f(v) = g(v)$.

Proposition 1. *If $G = (V, E)$ is given with a nice tree decomposition of width at most k , and with $O(n)$ nodes, then the 3-coloring problem on G can be solved in $O(3^k n)$ time.*

Proof. We discuss for each of the four types of nodes: **leaf**, **introduce**, **forget**, **join** how the table A_i can be computed, given such tables of the children of i , in $O(3^k)$ time. The algorithm then is as follows: in postorder, we compute for each node of the nice tree decomposition the table A_i . In $O(3^k n)$ time we thus have the table A_r for the root r of the nice tree decomposition. Finally, note that G_r equals G , and thus, G has a 3-coloring, if and only if at least one entry in A_r is true. So, we end the algorithm by inspecting A_r for a value true.

Computing A_i for a **leaf** node i is trivial. Recall that G_i just has one vertex and no edges; each of the three possible colorings of this vertex corresponds to a true entry in the table A_i .

Suppose i is an **introduce** node with child j with $X_i = X_j \cup \{v\}$. Consider a coloring $c : X_i \rightarrow \{1, 2, 3\}$. Let c' be the restriction of c to X_j . It is not difficult to see that we have:

$$A_i(c) \Leftrightarrow A_j(c') \wedge \forall w \in X_j : \{v, w\} \notin E \vee c(v) \neq c(w).$$

Suppose i is a **forget** node with child j with $X_i = X_j - \{v\}$. Now we have for all colorings $c : X_i \rightarrow \{1, 2, 3\}$, that $A_i(c)$ is true, iff there is a coloring c' of X_j with $A_j(c')$ true and c is the restriction of c' .

For a **join** node i with children j' and j'' , we have for each $c : X_i \rightarrow \{1, 2, 3\}$, that $A_i(c) = A_{j'}(c) \wedge A_{j''}(c)$.

Correctness can easily be derived. In each case, the way G_i is obtained from the graphs associated with the children of i is used; see the discussion in Section 2. It is also easy to see that the time to compute a table is linear in its size. \square

Designing an algorithm of the type given above follows a number of steps:

- What information should be stored at a table of a node? This information characterizes the subgraph G_i . Often, the notion of a *partial solution* is used; each partial solution has a characterization, and we tabulate the different characterizations. In case of optimization problems, one can assign costs to partial solutions, and then tabulate for each characteristic the minimum or maximum cost of a partial solution with this characteristic. For an example of the latter, see our discussion of the DOMINATING SET problem. In the case of the 3-COLORING problem, a 3-coloring of G_i is a partial solution, which is characterized by the colors given to the vertices in X_i . A value true implies that there is a partial solution with this characteristic.
- Design for each of the four types of nodes (**leaf**, **introduce**, **forget**, **join**), an algorithm that computes the table for the node, given the tables of the children.
- Show that the answer to the problem can be derived from the table for the root r , using that $G = G_r$.

The second step is not always necessary: Fellows and Langston [65] introduce the *Myhill-Nerode* perspective of algorithms on tree decompositions. We discuss this briefly in the next section.

3.2 Dynamic Programming and Finite State Automata

In this section, we look at the algorithm from a perspective, first introduced by Fellows and Langston [65], namely, we view the algorithm as running on a finite state automaton or finite state tree automaton. For an easier exposition, we consider the algorithm as running on a path decomposition of bounded width. The discussion can be extended to tree decompositions. When using path decompositions, our algorithm corresponds to a finite state automaton; when using tree decompositions, this generalizes to a *finite state tree automaton*.

Consider the algorithm that was given in the previous section. We assume it runs on a path decomposition of width k , with k a constant; i.e., we do

not have **join** nodes. For each node in the path decomposition, we computed a table. To denote a table, we need a constant number of bits, i.e., there the number of possible tables is a constant (only depending on the width of the path decomposition.)

As discussed earlier, we can represent a nice path decomposition of width k by a string in an alphabet whose size is bounded by a function of k ($2k + k + 1$). Now, for a bag, the table that is computed by the algorithm for that bag only depends on the table of the previous bag, whether the bag is an **introduce** or **forget** node, and which vertex is forgotten, or what incidences there are to the **introduce** node. Thus, the table depends on the previous table and the 'character' of the bag.

Thus, we can view the algorithm as a finite state automaton: each possible table corresponds to a state of the automaton, and the next state only depends on the previous state and the character. The table for the last bag decides if the input is accepted or rejected.

Many dynamic programming algorithms on path decompositions can be seen as finite state automata: the main ingredients are that tables must have a number of bits that is a function of the width, and that tables only depend on the previous table and the type of bag, as discussed above. Algorithms on tree decompositions can be viewed in a similar way as *finite state tree automata*.

This way of viewing algorithms as automata has important consequences: several classic results of automata theory can be used. For instance, it is decidable whether two finite state automata recognize the same set of strings, and thus, if we have two dynamic programming algorithms of the proper form, we can determine if these give the same output for all graphs of pathwidth at most k . Some corollaries of this will be discussed later.

3.3 Finite Index

When designing dynamic programming algorithms for problems on graphs, usually the first step ("what should we store in tables") is the most important. When tables have a constant number of bits, this step gives us equivalence relations on k -terminal graphs (for each k).

Suppose we have a decision problem Q on graphs. Let $\sim_{Q,k}$ be the equivalence relation on k -terminal graphs, with for all k -terminal graphs G, H , $G \sim_{Q,k} H$, if and only if for all k -terminal graphs K , $Q(G \oplus K)$, if and only if $Q(H \oplus K)$.

Suppose we have a dynamic programming algorithm A , that runs in $f(k)n$ time when given a tree decomposition of width k , and each table has $O(1)$ bits. Let $\sim_{A,k}$ be the equivalence relation on k -terminal graphs, with $G \sim_{A,k} H$ if when the table that is computed by A when G is the k -terminal subgraph associated with a bag equals the table that is computed when H is the k -terminal subgraph associated with a bag. Now, by closely observing the working of the dynamic programming algorithm, one can observe that the output of A will be the same for $G \oplus K$ as for $H \oplus K$, for any K , and thus $\sim_{A,k}$ is a refinement of $\sim_{Q,k}$.

We say that Q is *finite index*, if for each k , $\sim_{Q,k}$ has a finite number of equivalence classes. Now, the famous Myhill-Nerode theorem for regular languages tells us that if Q is finite index, then Q is regular. In particular, the theorem tells us that when we have established an equivalence relation that is (a refinement of) $\sim_{Q,\ell}$ for all $\ell \leq k$, then from this, we can derive the finite state automaton, i.e., a dynamic programming algorithm for graphs of pathwidth at most k . As the Myhill-Nerode theorem also holds for tree automata, we obtain the same result for graphs of treewidth at most k .

This has two consequences: it confirms the intuition that the design of the equivalence relation is the important step in the design of the algorithms that run on path or tree decompositions, and it allows us to avoid in several cases the design by hand of the procedures that tell how to compute tables for **join**, **introduce** and **forget** nodes.

3.4 Courcelle's Theorem

As said, for many problems, linear-time algorithms have been found for the problems restricted to graphs of bounded treewidth. Often, constructing such algorithms means to pay attention to many details. Fortunately, there are also algorithmic meta-theorems, that allow us to establish for a large number of problems the existence of linear-time algorithms when restricted to graphs of bounded treewidth. By far the most important of these algorithmic meta-theorems is *Courcelle's theorem*.

Theorem 4 (Courcelle [42]). *For each problem P , that can be formulated in Counting Monadic Second Order Logic, there exists an algorithm that decides P on a given graph G , and that uses linear time for graphs of treewidth bounded by some constant.*

Counting Monadic Second Order Logic (CMSOL) is a language in which we can express properties of graphs. The simpler version of Monadic Second Order Logic (MSOL) has the following elements: tests if a vertex is incident with an edge ($v \in e$), tests if two vertices are adjacent ($\{v, w\} \in E$), tests if a vertex (edge) is an element of a vertex (edge) set ($v \in W$, $e \in F$), Boolean operations (\neg , \vee , \wedge , \Rightarrow , \dots), equality of variables, quantification over vertices and edges ($\exists v \in V$, $\exists e \in E$, $\forall v \in V$, $\forall e \in E$), and quantification over vertex and edge sets ($\exists W \subseteq V$, $\exists F \subseteq E$, $\forall W \subseteq V$, $\forall F \subseteq E$). CMSOL has in addition operations that decide if the size of a set modulus some constant equals another constant, i.e., for constants c_1 and c_2 , the language has expressions $|W| \bmod c_1 = c_2$ and $|F| \bmod c_1 = c_2$.

For example, the property that G is bipartite can be expressed as:

$$\exists W \subseteq V : \forall e \in E : \exists v \in V : \exists w \in V : v \in e \wedge w \in e \wedge v \in W \wedge \neg(w \in W)$$

Many well known and important graph properties, including many NP-hard properties, can be expressed in CMSOL. Besides an alternative proof of Courcelle's theorem, Borie et al. [32] show how to express many graph properties in CMSOL. See also [92] for a different proof that gives better constant factors.

Several extensions to Courcelle's theorem have been found. An important one allows us to obtain linear-time algorithms for many optimization problems restricted to graphs of bounded treewidth. Consider a CMSOL property P with one free vertex or edge set variable. The problems to find a minimum size set of vertices W or edges F such that $P(W)$ or $P(F)$ holds can also be solved in linear time for graphs of bounded treewidth; the same holds when we want to find such a set of maximum weight, or if weighted variants are considered. See e.g., [6, 32, 31, 33, 45].

Another important variant is the result by Courcelle et al. [44] who show that a similar result holds for graphs of bounded cliquewidth for CMSOL without edge set quantifications. As bounded cliquewidth is equivalent (with different bounds) to bounded NLC-width, bounded rankwidth, or bounded booleanwidth, we have for each of these graph measures many problems that can be solved in linear or polynomial time when they have bounded 'width'.

The 'automaton view' also helps to see another result by Courcelle: for each graph property P in CMSOL and integer k , it is decidable if all (or no) graphs of treewidth (or pathwidth) at most k fulfill property P . The main idea of the proof (sketched here for the case of pathwidth) is the following: build the finite state automaton for path (tree) decompositions of width at most k . Also, build the finite state automaton that checks if a sequence of bag types represents a possible path decomposition (cf. the discussion in Section 2). Now use Myhill-Nerode theory to check if these two automata accept the same set of strings.

Theorem 5. *Let P be a property in CMSOL, and k be an integer. It is decidable whether all graphs of pathwidth (treewidth) at most k have property P , and whether no graphs of pathwidth (treewidth) at most k have property P .*

3.5 Courcelle's Conjecture

Courcelle's theorem (Theorem 4) shows that expressibility in CMSOL implies finite index. Courcelle conjectured that the reverse also holds. (See also e.g., [43].) Proofs of the conjecture for special cases were obtained by Kabanets [86] (graphs of bounded pathwidth) and Kaller [87] (graphs of treewidth 3 and k -connected graphs of treewidth k). In 1998, Lapoire [98] announced a proof for the conjecture, but a refereed full version of the proof has not been published.

3.6 Running Times as Function of Pathwidth and Treewidth

For many problems, Courcelle's theorem gives a relatively fast way of establishing that the problem is fixed-parameter tractable with respect to treewidth, i.e., that there is an algorithm that solves the problem in linear time for graphs of bounded treewidth. The constant factors of such algorithms will however be large, and better constant factors can often be obtained when designing tailor-made algorithms for specific problems.

For some problems, the running time can be improved with help of additional techniques. One of these was introduced by van Rooij et al. [142], see also [30].

Here, a generalization of fast subset convolution is used to speed up algorithms on tree decompositions, in particular the **join** operation. The main idea is the following: the information stored in a table in the dynamic programming algorithm can often be represented in different ways. Some of these allow for a faster **join** operation, while others allow for faster **introduce** (or **forget**). With help of fast subset convolution or generalizations of it, one can quickly transform a table in one representation to its equivalent table in the other representation. Tables are again computed in postorder, i.e., bottom-up, but when necessary, the representation is changed.

Very recently, Cygan et al. [46] introduced a new technique that speeds up several computations on tree decompositions, which they call *Cut and Count*. Here, algorithms on tree decompositions are made faster by using a *randomized* approach. In this way, Cygan et al. [46] obtain randomized algorithms whose dependence on the width of the given tree decomposition is only single exponential, (i.e., of the form $O^*(c^k)$ for some constant c) while the known ‘classic’ dynamic programming algorithms have a running time $\Theta^*(2^{k \log k})$ or worse for these problems.

3.7 Lower Bounds

For a number of problems, there are also lower bounds known (for the dependency of the running time on the treewidth). Lokshantov et al. [104] have shown such lower bounds for a large number of problems. For instance, consider the 3-COLORING problem. We have seen that this problem can be solved in $O(3^k n)$ time; Lokshantov et al. [104] prove that there exists no algorithm that uses $(3 - \epsilon)^{tw(G)} n^{O(1)}$ time for any $\epsilon > 0$, unless the Strong Exponential Time Hypothesis [84, 48] does not hold. Other problems where the known upper bound matches this type of lower bound include DOMINATING SET, q -COLORING for constant q ; INDEPENDENT SET. See also [46].

3.8 Special Classes of Graphs

Efficient algorithms for graphs of bounded treewidth can also help to obtain fast(er) algorithms for problems on special types of graphs. Two important examples of this are the *planar* graphs and *graphs of bounded degree*.

Planar graphs have treewidth $O(\sqrt{n})$. The fact can be shown to follow from the Lipton-Tarjan planar separator theorem [102, 103]; and vice versa, the planar separator theorem can be obtained as corollary from the fact that planar graphs have treewidth $O(\sqrt{n})$, see [16]. Fomin and Thilikos [74] showed that the treewidth of a planar graph is bounded by $3.182\sqrt{n}$, and also showed that the branchwidth of a planar graph is at most $2.122\sqrt{n}$.

As a consequence, for many graph problems, there are $O(c^{\sqrt{n}})$ algorithms, and sometimes $O(c^{\sqrt{n} \log n})$ time algorithms when the inputs are restricted to planar graphs. An example is 3-COLORING, see Section 3.1.

For several problems, dynamic programming as discussed above leads to algorithms that use $O(c^{\sqrt{n} \log n})$ time. With help of additional arguments, algorithms

that use $O(c^{\sqrt{n} \log n})$ time can be obtained for several problems on planar graphs, like HAMILTONIAN CIRCUIT. One can either exploit planarity (leading to an analysis with Catalan structures), see e.g., [53, 52]; or use the probabilistic approach by Cygan et al. [46] which was discussed above. Other algorithms for planar graphs that exploit treewidth (or the related notion of branchwidth) can be found in e.g., [74, 95, 136].

For graphs of bounded degree, we have the following theorem by Fomin et. al. [71].

Theorem 6 (Fomin et al. [71]). *For $\epsilon > 0$, there exists an n_ϵ , such that for all graphs G with $n \geq n_\epsilon$ vertices of which n_3 have degree 3, n_4 have degree 4, n_5 have degree 5, n_6 have degree 6, and $n_{>6}$ have degree more than 6, the pathwidth of G is at most*

$$\frac{1}{6}n_3 + \frac{1}{3}n_4 + \frac{13}{30}n_5 + \frac{23}{45}n_6 + n_{>6} + \epsilon \cdot n$$

The result can in several cases be used to obtain faster exact (exponential time) algorithms for graph problems (i.e., ‘problems parameterized by the number of vertices n ’), see e.g., [71, 141]. Kneis et al. [93] showed that graphs have pathwidth at most $m/5.769 + O(\log n)$, m the number of edges. This also has several algorithmic consequences, e.g., faster exact algorithms for sparse graphs for MAX CUT and for MAX 2SAT.

4 Graph Minors

In this section, we briefly review a few results from graph minor theory, with some emphasis on its role for the theory of treewidth and related notions. For more extensive overviews, see e.g., [10, 75, 112], or [54, Chapter 7].

In a long series of papers [110, 113, 111, 117, 114–116, 119, 118, 120, 122–126, 128, 127, 129–132, 121, 133], Robertson and Seymour obtained a number of important and fundamental results on graph minors. The central result is the graph minor theorem. (A graph G is minor minimal in a set of graphs if no other graph in the set is a minor of it. Isomorphic graphs are considered to be identical.)

Theorem 7 (Robertson and Seymour). *Any set of graphs has a finite number of minor-minimal elements.*

Equivalent to Theorem 7 is the following.

Theorem 8 (Robertson and Seymour [128]). *Let \mathcal{G} be a collection of graphs that is closed under taking minors. Then there exists a finite set $ob(\mathcal{G})$, called the obstruction set of \mathcal{G} , such that for each graph G , we have that $G \in \mathcal{G}$, if and only if there is no graph $H \in ob(\mathcal{G})$ that is a minor of G .*

Theorem 8 has important algorithmic consequences. Several such results were established in the 1980’s and 1990’s by Fellows and Langston, see e.g., [62, 64,

66, 63, 60, 67, 68]. As also for fixed graphs H , testing if H is a minor can be done in $O(n^3)$ time [124], there exists for each set of graphs that is closed under taking of minors an $O(n^3)$ time membership test. This result however is non-constructive: we know that the algorithm exists but do not know the algorithm itself, as we may not know the obstruction set.

For graphs of bounded treewidth, faster algorithms exist: for a fixed H and fixed integer ℓ , there is a (dynamic programming) algorithm that tests in linear time whether H is a minor of an input graph G , given with a tree decomposition of width at most ℓ . Combined with the result discussed in Section 5.6, we have that each class of graphs that is closed under minors and has bounded treewidth can be recognized in linear time. Then, we use the following result.

Theorem 9 (Robertson et al. [114, 134]). *For each planar graph H , there is a constant c_H , such that each graph G that does not have H as a minor has treewidth at most c_H .*

(A similar result bounds the pathwidth of graphs that do not have some fixed forest H as a minor [110, 11].) Thus, any minor closed class of graphs that does not include all planar graphs has a linear-time recognition algorithm. This result, however, is again non-constructive.

Theorem 10. *Let \mathcal{G} be a class of graphs that is closed under taking of minors. Suppose we can construct a dynamic programming algorithm on tree decompositions of bounded width, that uses $O(1)$ bits per table/node for the problem to recognize graphs in \mathcal{G} . Suppose an integer k is known such that all graphs in \mathcal{G} have treewidth at most k . Then the obstruction set of \mathcal{G} is computable.*

Proof. The result follows from the Myhill-Nerode perspective, as discussed in Section 3.2, as introduced by Fellows and Langston [65]. The dynamic programming algorithm on tree decompositions corresponds to a finite state tree automaton. For each finite set of graphs \mathcal{Z} , the property that an input graph G has no graph from \mathcal{Z} as a minor can be formulated in monadic second order logic (see e.g., the discussion in [32]) and thus, by Courcelle's theorem (Theorem 4), we can construct a tree automaton that gets as input the representation of a nice tree decomposition of width at most k , and tests whether G has a tree decomposition of width at most k . We now can decide, using a tree automaton equivalent of the classic Myhill Nerode theorem for finite state automata, whether the two machines accept the same language. Thus, for each finite set of graphs, we can check if this is the obstruction set of \mathcal{G} . By enumerating all finite sets of graphs, we eventually find the obstruction set. \square

See also e.g., [37].

5 Deciding Treewidth and Pathwidth

In this section, we discuss the problems, for fixed integers k , to decide for a given graph $G = (V, E)$ whether its treewidth is at most k . We also look at the

constructive variant: if the answer is yes, the algorithm also has to output a tree decomposition of width at most k , and we consider the variants of this problem where we consider pathwidth and path decompositions instead.

The problem to determine for a given graph G and integer k , whether the treewidth of G is at most k is NP-complete [4]. The NP-completeness proof of Arnborg et al. [4] shows that treewidth is NP-complete for co-bipartite graphs, i.e., graphs that are obtained by adding some edges between vertices in two cliques. They also show that for these graphs, the treewidth equals the pathwidth, and thus obtain also the NP-completeness of pathwidth. An independent NP-completeness proof of pathwidth (or, more precisely, for a notion equivalent to pathwidth) was found by Lengauer [101].

In the remainder of this section, we consider the fixed-parameter cases for TREEWIDTH and PATHWIDTH.

5.1 Membership in XP

Downey and Fellows [54] define the class XP, as the class of parameterized problems that are solvable in time $O(n^{f(k)})$ for some function f .

The result that TREEWIDTH belongs to XP dates from far before the terminology. In the 1980s, Arnborg et al. [4] give a clever dynamic programming algorithm for TREEWIDTH that uses $O(n^{k+2})$ time. The first algorithm whose running time is in XP for PATHWIDTH was found by Ellis et al.; this complicated algorithm (formulated on the equivalent notion of vertex separation number) only appears in a technical report in 1987 [58]. Both algorithms solve the constructive versions of the problem, i.e., they also give tree or path decompositions of width at most k , if existing.

5.2 Nonconstructive Advances

The fact that TREEWIDTH is fixed-parameter tractable was first obtained as a consequence from the work of Robertson and Seymour on graph minors. We briefly discuss how the results discussed in Section 4 show that TREEWIDTH and PATHWIDTH are (non uniform) fixed-parameter tractable. We use PATHWIDTH as running example.

Lemma 2. *For each fixed k , the class of graphs with pathwidth at most k is closed under minor taking.*

Proof. Suppose H is a minor of G , and G has pathwidth at most k . Consider a path decomposition of G of width at most k . Consider the sequence of operations that shows that H is a minor of G . For a deletion of a vertex v , we remove v from all bags of the path decomposition. For a deletion of an edge, we do nothing. For the contraction of an edge $\{v, w\}$ to a vertex x , we replace each occurrence of v and/or w in a bag by an occurrence of x . As a result, we obtain a path decomposition of H of the same or smaller width. \square

Thus we have by the results in Section 4:

Proposition 2. *For each fixed k , there is an $O(n^3)$ time algorithm, that given a graph, tests if G has pathwidth at most k .*

The result can be improved in several ways: the cubic time can be brought back to linear time. But also: this algorithm is *non-uniform*, and *non-constructive* in two ways: it does not provide a corresponding path decomposition, and we do not have the algorithm itself: as the proof of Theorem 8 is non-constructive, we know that the obstruction set and thus the algorithm exists, but we do not know this set and thus this algorithm (so far). In later parts of this section, we will overcome these points.

To speed up the algorithm, we can use the fact that the treewidth of graphs is bounded by its pathwidth, and that we can formulate for each fixed graph H , the property that H is a minor of a given graph $G = (V, E)$ in monadic second order logic. Thus, by Courcelle's theorem (see Section 3.4), we have that for fixed k , there exist a linear-time algorithm, that given a tree or path decomposition of bounded width of the input graph G , tests if the pathwidth of G is at most k , by verifying whether G contains any of the graphs from the obstruction set of graphs of pathwidth at most k as a minor.

To find such a tree or path decomposition, one could either use an approximation algorithm for treewidth (or pathwidth); such an algorithm should use time that is polynomial in n but can be exponential in k . The first such algorithm was given in terms of branchwidth and branch decompositions by Robertson and Seymour in [124]: this algorithm finds in $O(3^{3k}n^2)$ time a branch decomposition of width $3k$. This result can easily be transferred to a similar result giving tree decompositions (with factor 4.5 instead of 3 for treewidth). Similar results with some improvements in bounds or running times were obtained by different authors, see e.g., [1, 51, 96, 109] or [90, Sec. 10.5]. Reed [109] obtained a running time of $O(n \log n)$.

Further speedup can be obtained with different methods, which will be discussed later.

5.3 Fighting Non-constructiveness: Self-reduction

One approach to overcome non-constructiveness is by the use of self-reduction. Fellows and Langston [69] (see also [36]) introduced a general technique to turn a non-constructive proof of the existence of an algorithm into a constructive one. We showcase the technique by using the pathwidth problem as example.

Self-reduction is a well known technique to turn algorithms for decision problems into algorithms for the constructive version of the problem: by running the decision algorithm multiple times on slightly modified inputs, the output for the constructive version is generated (e.g., we construct a certificate for a problem in NP.) In the approach of Fellows and Langston, the technique is taken one step further: besides constructing the certificate (in this case, a path decomposition of width at most k , or, equivalently, an interval supergraph with maximum clique size at most $k + 1$), but we also circumvent the fact that we do not know the obstruction set in advance.

First, suppose we have a decision algorithm A for the problem to test for a given graph G if the pathwidth of G is at most k . First, run A on G . If A tells that the pathwidth of G is more than k , we halt. Otherwise, we use $O(n^2)$ runs of A to build an interval graph H with maximum clique size k . (Recall Theorem 3.) Take an auxiliary graph H , which we initially set to be equal to G . Now, for each pair of disjoint vertices $v, w \in V$, $\{v, w\} \notin E$, we test if the pathwidth of the graph, obtained by adding $\{v, w\}$ to H is at most k . If so, we add the edge $\{v, w\}$ to H . Call this algorithm B . The output of algorithm B is a maximal supergraph H of G that has pathwidth at most k ; more specifically, this graph H is an interval graph with maximum clique size $k + 1$.

Suppose we have a set of graphs X that is a subset of the obstruction set of the graphs of pathwidth at most k . We build an algorithm C that, given a graph G , either decides that the pathwidth of G is at most k , or gives a path decomposition of G of width at most k , or decides that X is a *proper* subset of the obstruction set of graphs of pathwidth at most k , as follows: run the following modification of algorithm B : instead of using A , we test if the input graph has a minor in X . If this algorithm tells that G has pathwidth more than k , then this is because a graph from X is a minor of G , and thus this is a correct output. Otherwise, we check if the output is indeed an interval graph with maximum clique size k . (This can be done in polynomial time, see e.g. [77].) If so, we are done; if not, we know that X was not equal to the obstruction set of graphs of pathwidth at most k .

We now can build an algorithm D , that given a graph G , either correctly decides that G has pathwidth more than k , or outputs a path decomposition of G of width at most k , as follows. Initially, let X be the empty set. Now, repeat the following step, until we are done. Enumerate all graphs G , and for each, test if G is not in X , and if G is a member of the obstruction set of graphs of pathwidth at most k , i.e., if the pathwidth of G is $k + 1$ and if each proper minor of G has pathwidth at most k . (We can use any algorithm for this.) If not, continue the enumeration of graphs. If the test succeeds, add G to X ; stop the enumeration of graphs, and run algorithm C with X . If algorithm C produces as output that G has pathwidth more than k , or a path decomposition of G , we are done; otherwise, we restart the enumeration of graphs, but now with the larger set X .

This is an fpt-algorithm, i.e., its running time is bounded by a function of k times a polynomial in n : we never have to enumerate graphs beyond the last graph in the obstruction set of pathwidth- k graphs, and thus X and the time for enumeration of graphs are bounded by a function of k . The algorithm is, of course, highly impractical, but showcases an important idea how we can turn non-constructive algorithms into constructive ones.

With some addition techniques, one can modify this algorithm such that it runs in $O(f(k)n^2)$ time, see [13]. The technique works for a large number of problems; see [69] for more details. For pathwidth, there exist more efficient algorithms, which will be discussed in later sections.

5.4 Graph Reduction Techniques

In this subsection, we discuss some algorithmic results for graphs of bounded treewidth that are based on the technique of graph reduction, now known under the name of *protrusions*. A simple example of this technique is the following algorithm that recognizes the graphs of treewidth at most one, i.e., the set of forests: while possible, remove vertices of degree one with their incident edge and vertices of degree zero. The empty graph results, if and only if the input graph was a forest.

If we add the reduction rule that removes vertices of degree two while adding an edge between their neighbors (if not already present), we obtain a recognition algorithm for graphs of treewidth at most two. Arnborg and Proskurowski gave a fast reduction algorithm for graphs of treewidth at most three [7], see also [107]. For treewidth 4, Sanders [135] found a linear-time recognition algorithm. An experimental evaluation of this algorithm by Hein and Koster [83] shows that this algorithm is practical.

In a more generalized setting, consider the equivalence relation $\sim_{Q,k}$ discussed in Section 3.2 for some decision problem Q on graphs. If we have k -terminal graphs G_1 and G_2 with $G_1 \sim_{Q,k} G_2$ and G_2 is smaller than G_1 , then this leads to the following algorithmic step: if we have G_1 as subgraph, with terminals of G_1 the only vertices in the subgraph with neighbors outside the subgraph, then we can replace G_1 by G_2 ; i.e., we transform $G = G_1 \oplus H$ to $G_2 \oplus H$. As $Q(G) = Q(G_2 \oplus H)$, the step is *safe*, as the answer to the problem at hand does not change.

A graph reduction algorithm can thus be based on a collection of such *safe reduction rules*. In 1993, Arnborg et al. [5] showed that for each fixed k , each graph problem that is finite state (and thus, including, all problems that can be formulated in monadic second order logic) there is a collection of reduction rules that give a linear time (on a random access machine with the uniform cost model) algorithm for graphs of treewidth at most k . Bodlaender and Hagerup [22] showed that one can obtain parallel algorithms based on graph reduction that use $O(\log n)$ time and $O(n)$ work; their version leads to linear-time sequential algorithms on the more standard pointer machine model. Bodlaender and van Antwerpen-de Fluiter [29, 49] showed that the technique can also be applied to some optimization problems (terming these *finite integer index*); a reduction rule not only changes the graph, but also adds a constant to one integer variable.

Graph reduction techniques are often used for preprocessing and kernelization. For the problem to determine the treewidth of a graph, graph reduction has been used in the setting of preprocessing [27, 56] and, recently, in the setting of kernelization [23]. Recently, graph reduction techniques were used to obtain kernelization results for other problems, including 'meta-kernelization' results: proofs that large collections of problems have kernelization algorithms when restricted to certain special graph classes (e.g., graphs embeddable on a fixed surface) [20, 72, 73].

A very recent (spring 2012) result by Drucker [55], combined with the kernelization lower bound techniques from [17], shows that TREEWIDTH (in its standard parameterization) does not have a polynomial kernel, unless $NP \subseteq coNP/poly$.

5.5 An Explicit Finite Congruence

The use of non-constructive methods can be avoided altogether by giving an explicit equivalence relation on path decompositions for certain types of subgraphs. The techniques can be generalized for treewidth and tree decompositions; we briefly discuss what additional technical difficulties are to be faced at the end of this section.

The results shown here were obtained by Bodlaender and Kloks [24] and Lagergren and Arnborg [97] in 1991; Fellows and Langston obtained similar results independently at the same time. Bodlaender et al. [19] discussed how such algorithms can be automatically be derived, and part of the discussion below is based on the ideas from [19].

Theorem 11. *Let k, ℓ be constants. There is (and we can explicitly describe) an algorithm, that given a graph $G = (V, E)$ with a path decomposition of G of width at most ℓ , decides if the pathwidth of G is at most k , and if so, finds a path decomposition of G of width at most k .*

Of course, we may assume that $k < \ell$, otherwise the problem is trivial.

We define a simple operation on sequences of integers, which we call *compacting*: if (a_1, \dots, a_q) is a sequence of integers, its *compacted* sequence is obtained by repeating the following step:

- If there are $i, j, j \geq i+2$, such that $a_i = \min_{i \leq i' \leq j} a_{i'}$ and $a_j = \max_{i \leq i' \leq j} a_{i'}$, then remove the numbers a_{i+1}, \dots, a_{j-1} from the sequence.
- If there are $i, j, j \geq i+2$, such that $a_i = \max_{i \leq i' \leq j} a_{i'}$ and $a_j = \min_{i \leq i' \leq j} a_{i'}$, then remove the numbers a_{i+1}, \dots, a_{j-1} from the sequence.

E.g., the compacted sequence of 3, 5, 7, 4, 2, 6 is 3, 7, 2, 6. The compacted sequence is unique, i.e., it does not depend on the order in which the steps are carried out.

Recall that pathwidth is equivalent to vertex separation number (Theorem 2.)

The *uncompacted fingerprint* of a linear ordering f of a terminal graph (V', E_i, X) is defined as follows. We partition f in *pieces* as follows: the first piece starts with the first vertex in the ordering, $f^{-1}(1)$. Now, visit the vertices from low to high number. Start a new piece when we see a terminal, i.e., a vertex in X , and start a new piece directly after a vertex that is the highest numbered neighbor of a vertex in X . We have for each piece an uncompacted fingerprint, and the uncompacted fingerprint of f is the sequence of uncompacted fingerprints of the pieces: Suppose we have the piece $f^{-1}(i), f^{-1}(i+1), \dots, f^{-1}(j)$. The first part of the uncompacted fingerprint is $X \cap \{f^{-1}(i)\}$, i.e., it tells whether the first vertex is a terminal and if so, what terminal; the second part is the sequence n_i, n_{i+1}, \dots, n_j , with $n_r = |\{w \in V \mid f(w) \leq r \wedge \exists \{w, x\} \in E : f(x) > r\}|$.

The *compacted fingerprint* is obtained by taking the uncompactd fingerprint and then compacting in each piece its sequence of numbers.

Lemma 3. *Let f and g be linear orderings of ℓ -terminal graph (V', E', X) , and let H be an ℓ -terminal graph, and $G = (V', E', X) \oplus H$. Let k be an integer.*

- (i). *Suppose f and g have the same uncompactd fingerprints. There exists a linear ordering of G with vertex separation number at most k that contains f as a subsequence, if and only if a linear ordering of G with vertex separation number at most k that contains g as a subsequence.*
- (ii). *Suppose f and g have the same compactd fingerprints. There exists a linear ordering of G with vertex separation number at most k that contains f as a subsequence, if and only if a linear ordering of G with vertex separation number at most k that contains g as a subsequence.*

The first part of the lemma is more or less trivial (except for an overload of terminology and notation). The second part contains the essential insight of the algorithms in [97, 24, 19]: the numbers that are forgotten when compacting are not essential when we need to determine if we can extend the ordering to an ordering of G of vertex separation number at most k .

Compactd sequences of integers in $\{0, \dots, k\}$ have length $O(k)$ [24], and thus for fixed ℓ , the number of compactd fingerprints of ℓ -terminal graphs is bounded by a constant.

The main idea of the algorithm of Theorem 11 is the following. Suppose we have a nice path decomposition (X_1, \dots, X_r) of width ℓ of G . For each i , we compute the set of compactd fingerprints of the linear orderings of the terminal graphs (V_i, E_i, X_i) of vertex separation number at most k . For **introduce** and for **forget** nodes, we have a subroutine that tells how such a set can be computed from the previous set. The pathwidth of G is at most k , if and only if the last of these sets (for (V_r, E_r, X_r)) is nonempty; note that $V = V_r$ and $E = E_r$. Similar as for many other dynamic programming algorithms, we can also construct (if existing) a corresponding linear ordering of width at most k , by going backwards through the tables. This linear ordering can easily be transformed to a path decomposition of width at most k (as in [89].)

Corollary 1. *For each k , the obstruction set of graphs of pathwidth at most k is computable.*

Proof. This follows directly from Theorem 10 and the discussion above. We have two ℓ -terminal subgraphs in the same equivalence class if they have the same set of fingerprints of linear orderings of vertex separation number at most k . \square

Similar results hold for treewidth. There are however several additional technical difficulties: the fingerprints are much harder to describe because of the tree structure, and a procedure has to be built for the **join** nodes. Similar results have been designed for other width parameters, e.g., [139, 140].

5.6 A Win-Win Theorem and a Linear-Time Algorithm

In this section, we sketch a linear-time algorithm for the fixed-parameter case of pathwidth. The algorithm follows the main ideas of the linear-time algorithm for the fixed-parameter case of treewidth [14]; some arguments are somewhat simpler for the case of pathwidth.

We denote with $G + \{v, w\}$ the graph obtained from G by adding the edge $\{v, w\}$. The following lemma is well known in its variant for treewidth, see e.g., [14]. Its statement and proof are identical for pathwidth.

Lemma 4. *Let $G = (V, E)$ be a graph, and $k \geq 0$. Suppose v and w have at least $k + 1$ common neighbors., Each path decomposition of width at most k of G is also a path decomposition of width at most k of $G + \{v, w\}$, and the pathwidth of G is at most k , if and only if the pathwidth of $G + \{v, w\}$ is at most k .*

Proof. Suppose v and w have at least $k + 1$ common neighbors.

Now, suppose that (X_1, \dots, X_r) is a path decomposition of G of width at most k . If there is a bag X_i with $v, w \in X_i$, then this is also a path decomposition of $G + \{v, w\}$ and hence the pathwidth of $G + \{v, w\}$ is at most k . Suppose such a bag does not exist. W.l.o.g., suppose the first bag that contains v is before the first bag that contains w . Let $v \in X_i$ with i maximal; and let $w \in X_j$ with j minimal. Now all common neighbors of v and w must belong to a bag containing v and to a bag containing w , and hence must belong to the first bag that contains w : this bag hence has size at least $k + 2$ as it contains w and at least $k + 1$ common neighbors of v and w , contradiction. The equivalence now follows from this, and the trivial observation that the pathwidth of G is never larger than the pathwidth of $G + \{v, w\}$. \square

Lemma 4 allows us to add edges between vertices with at least $k + 1$ common neighbors, without changing the answer to the question if the graph at hand has pathwidth at most k . In order to get a linear-time algorithm, we only look at neighbors of bounded degree. In this case, we define a number b_k and use it as upper bound for the degree of neighbors to make the proof work.

Define $a_k = \frac{k}{2}(2k + 2)(2k + 1) + k + 2$, and $b_k = a_k + k + 1$.

The k -improved graph of a graph $G = (V, E)$ is obtained from G by adding an edge between each pair of nonadjacent vertices v, w such that there are at least $k + 1$ vertices of degree at most b_k that are a common neighbor of v and w .

Building the k -improved graph is *not* an iterative process: the new edges are added simultaneously for all pairs in one round. It is well possible that the k -improved graph of the k -improved graph of G has more edges than the k -improved graph of G , but taking the closure of the improvement operation might take too much time.

Suppose we have a graph $G = (V, E_G)$ and its k -improved graph $H = (V, E_H)$. We say that a vertex is i -simplicial, if its neighbors in G form a clique in H , i.e., for each pair of edges $\{v, w\} \in E_G$, $\{v, x\} \in E_G$, we have $w = x$ or $\{w, x\} \in E_H$.

The following theorem gives us a ‘win-win’ approach to computing treewidth: we first make the improved graph; then greedily compute some maximal matching M . The theorem shows that we either have ‘a large maximal matching’ or

'many simplicial vertices'; in both cases, we can solve the problem by first solving the problem on a graph with linearly many fewer vertices and then running the algorithm that was discussed in Section 5.5. (For other win-win theorems, see e.g., [66],[54, Chapter 8.1].)

Theorem 12. *Let $G = (V, E)$ be a graph of pathwidth at most k . Let H be the k -improved graph of G . Let M be a maximal matching in H . Let X be the set of i -simplicial vertices in G . Then $2|M| + |X| \geq \lfloor n/a_k \rfloor$.*

Proof. By Lemma 4, the pathwidth of H is at most k . Consider a nice path decomposition (X_1, \dots, X_r) of H (and hence also of G) of width at most k . This path decomposition has n **introduce** nodes or **leaf** nodes: each vertex is introduced exactly once (with one vertex introduced in X_1). A *piece* of the path decomposition is a collection of successive nodes that contains exactly a_k introduce nodes. Note that the path decomposition contains $\lfloor n/a_k \rfloor$ non-overlapping pieces. A central part of the proof is the following claim.

Claim. Let $(X_i, X_{i+1}, \dots, X_j)$ be a piece. Let $W = \bigcup_{s=i+1}^{j-1} X_s - (X_i \cup X_j)$. W contains a vertex that is an endpoint of an edge in M or that is i -simplicial.

Proof. Let W' be the set of vertices that are 'introduced' by an **introduce** node in the piece. As we have a_k **introduce** nodes in the piece, $|W'| = a_k$. Vertices in X_i are introduced in a node with index at most i , so $W = W' - X_j$, and hence $W \geq a_k - (k + 1) = \frac{k}{2}(2k + 2)(2k + 1) + 1$.

Consider a vertex $v \in W$. If v is i -simplicial, then the claim holds, so suppose v is not i -simplicial. Thus, v must have two neighbors in G that are not adjacent in H , say x and y . As v belongs to a bag $X_{i'}$ with $i < i' < j$, but v does not belong to X_i or X_j , the only bags v can belong to are the bags $X_{i''}$ with $i < i'' < j$, and hence all neighbors of v belong to $W \cup X_i \cup X_j$, and hence v has degree at most $a_k + k + 1 = b_k$.

First, suppose $x \in W$. Then either v is an endpoint of an edge in M , x is an endpoint of an edge in M , or M is not a maximal matching. So, the claim holds in this case. Similarly if $y \in M$. The case that remains is that both x and y belong to $X_i \cup X_j$.

I.e., we have that each vertex in W has two nonadjacent neighbors in $X_i \cup X_j$, and degree at most b_k . As there are at most $\frac{1}{2}(2k + 2)(2k + 1)$ pairs of vertices in $X_i \cup X_j$, there must be a pair of nonadjacent vertices in $X_i \cup X_j$ that has at least $k + 1$ common neighbors in W , each with of degree at most b_k . But then the edge $\{v, w\}$ must have been added during the improvement step, i.e., $\{v, w\} \in E_H$, contradiction. □

The proof of Theorem 12 can now easily be concluded: we have $\lfloor n/a_k \rfloor$ nonoverlapping pieces. Each of these pieces contains a vertex that is i -simplicial or endpoint of an edge in the matching M . As these vertices never belong to the first or last bag of a piece, none of these vertices can belong to two or more pieces, and hence we have $\lfloor n/a_k \rfloor$ vertices that are i -simplicial or endpoint of an edge in M , which implies that $2|M| + |X| \geq \lfloor n/a_k \rfloor$. □

We now sketch the linear-time algorithm. The algorithm gets as input a graph $G = (V, E)$, and either outputs **no** (the pathwidth of G is more than k), or outputs a path decomposition of width at most k of G . It uses the algorithm of Section 5.5 as a subroutine. Correctness and running time will be argued later.

- (i). If G has at most $3a_k$ vertices, then solve the pathwidth problem by any deterministic algorithm, e.g., [4].
- (ii). Compute the k -improved graph $H = (V, E_H)$.
- (iii). Compute a maximal matching M in H .
- (iv). Compute the set X of i -simplicial vertices of degree at most k in H .
- (v). If $2|M| + |X| < \lfloor n/a_k \rfloor$ then output **no**.
- (vi). If $|X| \geq |M|$ then
 - (a) Let H' be obtained from H by removing all vertices in X from H .
 - (b) Recursively call the algorithm on H' .
 - (c) If the pathwidth of H' is larger than k , then output **no**.
 - (d) Otherwise, transform the path decomposition of H' of width at most k to a path decomposition of width at most $k + 1$ of H .
- (vii). Else ($|X| < |M|$)
 - (a) Let H'' be obtained from H by contracting all edges in M .
 - (b) Recursively call the algorithm on H'' .
 - (c) If the pathwidth of H'' is larger than k , then output **no**.
 - (d) Otherwise, transform the path decomposition of H'' of width at most k to a path decomposition of width at most $2k + 1$ of H .
- (viii). (Now, we have a path decomposition of H of width at most $2k + 1$.) Use the algorithm of Section 5.5 on H using the path decomposition constructed in the earlier step.

Several of the steps need more detail, and a proof that they can be performed in linear time. First, we argue correctness of the algorithm. We first consider Step 3. If the pathwidth of G is at most k , then the pathwidth of H is at most k (Lemma 4). Thus H has no clique of size $k + 1$ or more, and hence there cannot be i -simplicial vertices of degree more than k . Thus, by Theorem 12, $2|M| + |X| \geq \lfloor n/a_k \rfloor$. So, if we decide **no** in Step 3, the pathwidth of G indeed is more than k . H' is a subgraph of H , so if H' has pathwidth more than k , then H and hence G has treewidth more than k . H'' is a minor of H , and as pathwidth cannot increase when taking minors (see Section 4), if the pathwidth of H'' is more than k , then the pathwidth of H and thus G is more than k .

We now discuss a few of the steps in more detail. Computing the k -improved graph can be done in linear time with help of the use of radix sort techniques (see [41, Chapter 8.3]). Take an initially empty multiset S . For each vertex v of degree at most b_k , insert each pair of neighbors of v in S . Radix sort S , and then detect which pairs appear at least $k + 1$ times. Add these pairs to G . By radix sorting the set of edges of G , we can remove parallel edges.

Computing i -simplicial vertices again needs to use of radix sort. We leave the details as an easy exercise.

For step (vi)(d), we must find for each i -simplicial vertex $v \in X$ a bag X_{i_v} in the path decomposition of H' that contains all neighbors of v . Such a bag exists,

by Lemma 1. To find the bags, one can either again exploit radix sort, or note that $v \in X_i$ with

$$i = \min_{w \in N_H(v)} \max\{j \mid w \in X_j\}$$

Add a bag with vertex set $X_{i_v} \cup \{v\}$, directly after X_{i_v} . (When more vertices are mapped to the same bag, we add a number of bags, each with one new vertex.)

Consider now step (vii)(d). For each edge $\{v, w\} \in M$, replace in each bag, each occurrence of the newly formed vertex by the contraction by v and w . In this way, bag sizes at most double, so the width is at most $2k + 1$.

We now can argue that the algorithm uses linear time. As $2|M| + |X| \geq \lfloor n/a_k \rfloor$, we have $|M| \geq \frac{1}{3} \lfloor n/a_k \rfloor$ or $|X| \geq \frac{1}{3} \lfloor n/a_k \rfloor$. So, when we recursively call the algorithm on H' or H'' , this graph has at most $(1 - \lfloor \frac{1}{3a_k} \rfloor)n$ vertices. So, the time of the algorithm on a graph with n vertices fulfills:

$$T(n) = T((1 - \frac{1}{3a_k})n) + O(n)$$

which implies $T(n) = O(n)$.

We now have shown the following result.

Theorem 13. *Let k be a constants. There is (and we can explicitly describe) an algorithm, that given a graph $G = (V, E)$ decides if the pathwidth of G is at most k , and if so, finds a path decomposition of G of width at most k .*

A generalization of the techniques shown above lead to a similar result for treewidth and tree decompositions [15].

6 Conclusions

In this paper, a number of results have been surveyed on algorithmic aspects of treewidth. There are still a large number of topics that have not been touched here, including most practical aspects of treewidth computations (see e.g., [25, 26]), computing treewidth on special graph classes (including the celebrated results of Bouchitté and Todinca on potential maximal cliques [34, 35]), the role of treewidth for bidimensionality theory, logspace algorithms [57], $W[1]$ -hardness proofs for some problems on graphs of bounded treewidth (e.g., [18, 61]), dynamic algorithms [81], and much much more. The area of algorithmic research of treewidth is a very lively one, but can already look back to a lively history with several intriguing aspects, like the special role on nonconstructive results.

I end with mentioning a few probably very hard challenges:

- What is the complexity of TREEWIDTH, restricted to *planar graphs*. For the related BRANCHWIDTH problem, the famous ratcatcher algorithm by Seymour and Thomas [137] solves it in polynomial time; for TREEWIDTH on planar graphs, neither a polynomial time algorithm nor an NP-completeness proof is known.

- Is it possible to approximate treewidth up to a constant factor? There is an approximation with ratio $O(\sqrt{\log n})$ [59], and it is easy to show that approximation with an additive constant term is not possible assuming $P \neq NP$ [21].
- An accessible proof for Courcelle’s conjecture, i.e., that shows that each problem that is finite index can be formulated in CMSOL.
- Is it possible to find an algorithm for TREEWIDTH that runs in $O(c^k n^{c'})$ for constants c and c' ? Perhaps a probabilistic algorithm using ideas from [46]?

Acknowledgment. I thank Mike Fellows and numerous other colleagues for collaboration, discussions, and inspiration! It is hard to be sufficiently complete for a survey like this one, and I apologize to those whose work was not but should have been mentioned here.

References

1. Amir, E.: Approximation algorithms for treewidth. *Algorithmica* 56, 448–479 (2010)
2. Andrzejak, A.: An algorithm for the Tutte polynomials of graphs of bounded treewidth. *Discrete Mathematics* 190, 39–54 (1998)
3. Arnborg, S.: Efficient algorithms for combinatorial problems on graphs with bounded decomposability – A survey. *BIT* 25, 2–23 (1985)
4. Arnborg, S., Corneil, D.G., Proskurowski, A.: Complexity of finding embeddings in a k -tree. *SIAM Journal on Algebraic and Discrete Methods* 8, 277–284 (1987)
5. Arnborg, S., Courcelle, B., Proskurowski, A., Seese, D.: An algebraic theory of graph reduction. *Journal of the ACM* 40, 1134–1164 (1993)
6. Arnborg, S., Lagergren, J., Seese, D.: Easy problems for tree-decomposable graphs. *Journal of Algorithms* 12, 308–340 (1991)
7. Arnborg, S., Proskurowski, A.: Characterization and recognition of partial 3-trees. *SIAM Journal on Algebraic and Discrete Methods* 7, 305–314 (1986)
8. Arnborg, S., Proskurowski, A.: Linear time algorithms for NP-hard problems restricted to partial k -trees. *Discrete Applied Mathematics* 23, 11–24 (1989)
9. Bern, M.W., Lawler, E.L., Wong, A.L.: Linear time computation of optimal subgraphs of decomposable graphs. *Journal of Algorithms* 8, 216–235 (1987)
10. Bienstock, D., Langston, M.A.: Algorithmic implications of the graph minor theorem. In: Ball, M.O., Magnanti, T.L., Monma, C.L., Nemhauser, G.L. (eds.) *Handbook of Operations Research and Management Science: Network Models*, pp. 481–502. North-Holland, Amsterdam (1995)
11. Bienstock, D., Robertson, N., Seymour, P.D., Thomas, R.: Quickly excluding a forest. *Journal of Combinatorial Theory, Series B* 52, 274–283 (1991)
12. Bodlaender, H.L.: Polynomial algorithms for graph isomorphism and chromatic index on partial k -trees. *Journal of Algorithms* 11, 631–643 (1990)
13. Bodlaender, H.L.: Improved self-reduction algorithms for graphs with bounded treewidth. *Discrete Applied Mathematics* 54, 101–115 (1994)
14. Bodlaender, H.L.: A linear time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal on Computing* 25, 1305–1317 (1996)

15. Bodlaender, H.L.: Treewidth: Algorithmic Techniques and Results. In: Privara, I., Ružička, P. (eds.) MFCS 1997. LNCS, vol. 1295, pp. 19–36. Springer, Heidelberg (1997)
16. Bodlaender, H.L.: A partial k -arboretum of graphs with bounded treewidth. *Theoretical Computer Science* 209, 1–45 (1998)
17. Bodlaender, H.L., Downey, R.G., Fellows, M.R., Hermelin, D.: On problems without polynomial kernels. *Journal of Computer and System Sciences* 75, 423–434 (2009)
18. Bodlaender, H.L., Fellows, M.R., Hallett, M.T., Wareham, H.T., Warnow, T.J.: The hardness of perfect phylogeny, feasible register assignment and other problems on thin colored graphs. *Theoretical Computer Science* 244, 167–188 (2000)
19. Bodlaender, H.L., Fellows, M.R., Thilikos, D.M.: Derivation of algorithms for cutwidth and related graph layout parameters. *Journal of Computer and System Sciences* 75, 231–244 (2009)
20. Bodlaender, H.L., Fomin, F.V., Lokshtanov, D., Penninkx, E., Saurabh, S., Thilikos, D.M.: (Meta) kernelization. In: *Proceedings of the 50th Annual Symposium on Foundations of Computer Science, FOCS 2009*, pp. 629–638. IEEE Computer Society (2009)
21. Bodlaender, H.L., Gilbert, J.R., Hafsteinsson, H., Kloks, T.: Approximating treewidth, pathwidth, frontsize, and minimum elimination tree height. *Journal of Algorithms* 18, 238–255 (1995)
22. Bodlaender, H.L., Hagerup, T.: Parallel algorithms with optimal speedup for bounded treewidth. *SIAM J. Comput.* 27, 1725–1746 (1998)
23. Bodlaender, H.L., Jansen, B.M.P., Kratsch, S.: Preprocessing for Treewidth: A Combinatorial Analysis through Kernelization. In: Aceto, L., Henzinger, M., Sgall, J. (eds.) ICALP 2011, Part I. LNCS, vol. 6755, pp. 437–448. Springer, Heidelberg (2011)
24. Bodlaender, H.L., Kloks, T.: Efficient and constructive algorithms for the pathwidth and treewidth of graphs. *Journal of Algorithms* 21, 358–402 (1996)
25. Bodlaender, H.L., Koster, A.M.C.A.: Treewidth computations I. Upper bounds. *Information and Computation* 208, 259–275 (2010)
26. Bodlaender, H.L., Koster, A.M.C.A.: Treewidth computations II. Lower bounds. *Information and Computation* 209, 1103–1119 (2011)
27. Bodlaender, H.L., Koster, A.M.C.A., Van den Eijkhof, F.: Pre-processing rules for triangulation of probabilistic networks. *Computational Intelligence* 21(3), 286–305 (2005)
28. Bodlaender, H.L., Möhring, R.H.: The pathwidth and treewidth of cographs. *SIAM Journal on Discrete Mathematics* 6, 181–188 (1993)
29. Bodlaender, H.L., van Antwerpen-de Fluiter, B.: Reduction algorithms for graphs of small treewidth. *Information and Computation* 167, 86–119 (2001)
30. Bodlaender, H.L., van Leeuwen, E.J., van Rooij, J.M.M., Vatshelle, M.: Faster Algorithms on Branch and Clique Decompositions. In: Hliněný, P., Kučera, A. (eds.) MFCS 2010. LNCS, vol. 6281, pp. 174–185. Springer, Heidelberg (2010)
31. Borie, R.B.: Generation of polynomial-time algorithms for some optimization problems on tree-decomposable graphs. *Algorithmica* 14, 123–137 (1995)
32. Borie, R.B., Parker, R.G., Tovey, C.A.: Automatic generation of linear-time algorithms from predicate calculus descriptions of problems on recursively constructed graph families. *Algorithmica* 7, 555–581 (1992)
33. Borie, R.B., Parker, R.G., Tovey, C.A.: Solving problems on recursively constructed graphs. *ACM Computing Surveys* 41(4) (2008)

34. Bouchitté, V., Todinca, I.: Treewidth and minimum fill-in: Grouping the minimal separators. *SIAM Journal on Computing* 31, 212–232 (2001)
35. Bouchitté, V., Todinca, I.: Listing all potential maximal cliques of a graph. *Theoretical Computer Science* 276, 17–32 (2002)
36. Brown, D.J., Fellows, M.R., Langston, M.A.: Polynomial-time self-reducibility: Theoretical motivations and practical results. *International Journal of Computer Mathematics* 31, 1–9 (1989)
37. Cattell, K., Dinneen, M.J., Downey, R.G., Fellows, M.R., Langston, M.A.: On computing graph minor obstruction sets. *Theoretical Computer Science* 233, 107–127 (2000)
38. Chaudhuri, S., Zaroliagis, C.D.: Shortest paths in digraphs of small treewidth. Part II: Optimal parallel algorithms. *Theoretical Computer Science* 203, 205–223 (1998)
39. Chaudhuri, S., Zaroliagis, C.D.: Shortest paths in digraphs of small treewidth. Part I: Sequential algorithms. *Algorithmica* 27, 212–226 (2000)
40. Chen, H.: Quantified constraint satisfaction and bounded treewidth. In: de Mántaras, R.L., Saitta, L. (eds.) *Proceedings of the 17th European Conference on Artificial Intelligence, ECAI 2004*, pp. 161–165 (2004)
41. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: *Introduction to Algorithms*, 2nd edn. MIT Press, Cambridge (2001)
42. Courcelle, B.: The monadic second-order logic of graphs I: Recognizable sets of finite graphs. *Information and Computation* 85, 12–75 (1990)
43. Courcelle, B.: The monadic second-order logic of graphs V: On closing the gap between definability and recognizability. *Theoretical Computer Science* 80, 153–202 (1991)
44. Courcelle, B., Makowsky, J.A., Rotics, U.: Linear time solvable optimization problems on graphs of bounded clique width. *Theoretical Computer Science* 33, 125–150 (2000)
45. Courcelle, B., Mosbah, M.: Monadic second-order evaluations on tree-decomposable graphs. *Theoretical Computer Science* 109, 49–82 (1993)
46. Cygan, M., Nederlof, J., Pilipczuk, M., Pilipczuk, M., van Rooij, J., Wojtaszczyk, J.O.: Solving connectivity problems parameterized by treewidth in single exponential time. In: *Proceedings of the 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011*, pp. 150–159 (2011)
47. Dalmau, V., Kolaitis, P.G., Vardi, M.Y.: Constraint Satisfaction, Bounded Treewidth, and Finite-Variable Logics. In: Van Hentenryck, P. (ed.) *CP 2002*. LNCS, vol. 2470, pp. 310–326. Springer, Heidelberg (2002)
48. Dantsin, E., Wolpert, A.: On Moderately Exponential Time for SAT. In: Strichman, O., Szeider, S. (eds.) *SAT 2010*. LNCS, vol. 6175, pp. 313–325. Springer, Heidelberg (2010)
49. de Fluiter, B.: *Algorithms for Graphs of Small Treewidth*. PhD thesis, Utrecht University (1997)
50. Díaz, J., Serna, M., Thilikos, D.M.: Counting H -colorings of partial k -trees. *Theoretical Computer Science* 281, 291–309 (2002)
51. Diestel, R., Jensen, T.R., Gorbunov, K.Y., Thomassen, C.: Highly connected sets and the excluded grid theorem. *Journal of Combinatorial Theory, Series B* 75, 61–73 (1999)
52. Dorn, F.: Dynamic programming and planarity: Improved tree-decomposition based algorithms. *Discrete Applied Mathematics* 158, 800–808 (2010)

53. Dorn, F., Penninkx, E., Bodlaender, H.L., Fomin, F.V.: Efficient exact algorithms on planar graphs: Exploiting sphere cut decompositions. *Algorithmica* 58, 790–810 (2010)
54. Downey, R.G., Fellows, M.R.: *Parameterized Complexity*. Springer (1999)
55. Drucker, A.: New limits to classical and quantum instance compression (preliminary draft) (2012) (manuscript)
56. van den Eijkhof, F., Bodlaender, H.L., Koster, A.M.C.A.: Safe reduction rules for weighted treewidth. *Algorithmica* 47, 138–158 (2007)
57. Elberfeld, M., Jakoby, A., Tantau, T.: Logspace versions of the theorems of Bodlaender and Courcelle. In: *Proceedings of the 51st Annual Symposium on Foundations of Computer Science, FOCS 2010*, pp. 143–152 (2010)
58. Ellis, J.A., Sudborough, I.H., Turner, J.: Graph separation and search number. Report DCS-66-IR, University of Victoria (1987)
59. Feige, U., Hajiaghayi, M., Lee, J.R.: Improved approximation algorithms for minimum weight vertex separators. *SIAM Journal on Computing* 38, 629–657 (2008)
60. Fellows, M.R.: The Robertson-Seymour theorems: A survey of applications. *Contemporary Mathematics* 89, 1–18 (1989)
61. Fellows, M.R., Fomin, F.V., Lokshtanov, D., Rosamond, F., Saurabh, S., Szeider, S., Thomassen, C.: On the complexity of some colorful problems parameterized by treewidth. *Information and Control* 209, 143–153 (2011)
62. Fellows, M.R., Langston, M.A.: Nonconstructive advances in polynomial-time complexity. *Information Processing Letters* 26, 157–162 (1987)
63. Fellows, M.R., Langston, M.A.: Fast Self-reduction Algorithms for Combinatorial Problems of VLSI Design. In: Reif, J.H. (ed.) *AWOC 1988*. LNCS, vol. 319, pp. 278–287. Springer, Heidelberg (1988)
64. Fellows, M.R., Langston, M.A.: Nonconstructive tools for proving polynomial-time decidability. *Journal of the ACM* 35, 727–739 (1988)
65. Fellows, M.R., Langston, M.A.: An analogue of the Myhill-Nerode theorem and its use in computing finite-basis characterizations. In: *Proceedings of the 30th Annual Symposium on Foundations of Computer Science, FOCS 1989*, pp. 520–525 (1989)
66. Fellows, M.R., Langston, M.A.: On search, decision and the efficiency of polynomial-time algorithms. In: *Proceedings of the 21st Annual Symposium on Theory of Computing, STOC 1989*, pp. 501–512 (1989)
67. Fellows, M.R., Langston, M.A.: Fast search algorithms for layout permutation problems. *International Journal on Computer Aided VLSI Design* 3, 325–340 (1991)
68. Fellows, M.R., Langston, M.A.: On well-partial-order theory and its application to combinatorial problems of VLSI design. *SIAM Journal on Discrete Mathematics* 5, 117–126 (1992)
69. Fellows, M.R., Langston, M.A.: On search, decision and the efficiency of polynomial-time algorithms. *Journal of Computer and System Sciences* 49, 769–779 (1994)
70. Fernández-Baca, D., Slutzki, G.: Parametric problems on graphs of bounded treewidth. *Journal of Algorithms* 16, 408–430 (1994)
71. Fomin, F.V., Gaspers, S., Saurabh, S., Stepanov, A.A.: On two techniques of combining branching and treewidth. *Algorithmica* 54, 181–207 (2009)
72. Fomin, F.V., Lokshtanov, D., Saurabh, S., Thilikos, D.M.: Bidimensionality and kernels. In: *Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010*, pp. 503–510 (2010)

73. Fomin, F.V., Lokshtanov, D., Saurabh, S., Thilikos, D.M.: Linear kernels for (connected) dominating set on H -minor-free graphs. In: Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, pp. 82–93 (2012)
74. Fomin, F.V., Thilikos, D.M.: New upper bounds on the decomposability of planar graphs. *Journal of Graph Theory* 51, 53–81 (2006)
75. Friedman, H., Robertson, N., Seymour, P.D.: The metamathematics of the graph minor theorem. *Contemporary Mathematics* 65, 229–261 (1987)
76. Ganian, R., Hliněný, P.: On parse trees and Myhill-Nerode-type tools for handling graphs of bounded rank-width. *Discrete Applied Mathematics* 158, 851–867 (2010)
77. Golumbic, M.C.: *Algorithmic Graph Theory and Perfect Graphs*. Academic Press, New York (1980)
78. Gottlob, G., Leone, N., Scarcello, F.: A comparison of structural CSP decomposition methods. *Acta Informatica* 124, 243–282 (2000)
79. Gupta, A., Nishimura, N.: The complexity of subgraph isomorphism for classes of partial k -trees. *Theoretical Computer Science* 164, 287–298 (1996)
80. Gustedt, J., Mæhle, O.A., Telle, J.A.: The Treewidth of Java Programs. In: Mount, D.M., Stein, C. (eds.) *ALENEX 2002*. LNCS, vol. 2409, pp. 86–97. Springer, Heidelberg (2002)
81. Hagerup, T.: Dynamic algorithms for graphs of bounded treewidth. *Algorithmica* 27, 292–315 (2000)
82. Hagerup, T., Katajainen, J., Nishimura, N., Ragde, P.: Characterizing multiterminal flow networks and computing flows in networks of small treewidth. *Journal of Computer and System Sciences* 57, 366–375 (1998)
83. Hein, A., Koster, A.M.C.A.: An Experimental Evaluation of Treewidth at Most Four Reductions. In: Pardalos, P.M., Rebennack, S. (eds.) *SEA 2011*. LNCS, vol. 6630, pp. 218–229. Springer, Heidelberg (2011)
84. Impagliazzo, R., Paturi, R.: On the complexity of k -SAT. *Journal of Computer and System Sciences* 62, 367–375 (2001)
85. Isobe, S., Zhou, X., Nishizeki, T.: A polynomial-time algorithm for finding total colorings of partial k -trees. *International Journal of Foundations of Computer Science* 10, 171–194 (1999)
86. Kabanets, V.: Recognizability Equals Definability for Partial k -Paths. In: Degano, P., Gorrieri, R., Marchetti-Spaccamela, A. (eds.) *ICALP 1997*. LNCS, vol. 1256, pp. 805–815. Springer, Heidelberg (1997)
87. Kaller, D.: Definability equals recognizability of partial 3-trees and k -connected partial k -trees. *Algorithmica* 27, 348–381 (2000)
88. Kashem, M.A., Zhou, X., Nishizeki, T.: Algorithms for generalized vertex-rankings of partial k -trees. *Theoretical Computer Science* 240, 407–427 (2000)
89. Kinnersley, N.G.: The vertex separation number of a graph equals its path width. *Information Processing Letters* 42, 345–350 (1992)
90. Kleinberg, J., Tardos, E.: *Algorithm Design*. Addison-Wesley, Boston (2005)
91. Kloks, T.: *Treewidth. Computations and Approximations*. LNCS, vol. 842. Springer, Heidelberg (1994)
92. Kneis, J., Langer, A., Rossmanith, P.: Courcelle’s theorem - a game-theoretic approach. *Discrete Optimization* 8(4), 568–594 (2011)
93. Kneis, J., Mölle, D., Richter, S., Rossmanith, P.: A bound on the pathwidth of sparse graphs with applications to exact algorithms 23, 407–427 (2009)
94. Koster, A.M.C.A., van Hoesel, S.P.M., Kolen, A.W.J.: Solving partial constraint satisfaction problems with tree decomposition. *Networks* 40(3), 170–180 (2002)

95. Koutsonas, A., Thilikos, D.M.: Planar feedback vertex set and face cover: combinatorial bounds and subexponential algorithms. *Algorithmica* 60, 987–1003 (2011)
96. Lagergren, J.: Efficient parallel algorithms for graphs of bounded tree-width. *Journal of Algorithms* 20, 20–44 (1996)
97. Lagergren, J., Arnborg, S.: Finding Minimal Forbidden Minors Using a Finite Congruence. In: Albert, J.L., Monien, B., Rodríguez-Artalejo, M. (eds.) *ICALP 1991*. LNCS, vol. 510, pp. 532–543. Springer, Heidelberg (1991)
98. Lapoire, D.: Recognizability Equals Definability, for Every Set of Graphs of Bounded Tree-width. In: Meinel, C., Morvan, M. (eds.) *STACS 1998*. LNCS, vol. 1373, pp. 618–628. Springer, Heidelberg (1998)
99. Lauritzen, S.J., Spiegelhalter, D.J.: Local computations with probabilities on graphical structures and their application to expert systems. *The Journal of the Royal Statistical Society, Series B (Methodological)* 50, 157–224 (1988)
100. Leaver-Fay, A., Liu, Y., Snoeyink, J.: Faster placement of hydrogen atoms in protein structures by dynamic programming. In: *Proceedings of the 6th Workshop on Algorithm Engineering and Experimentation and the 1st Workshop on Analytic Algorithmics and Combinatorics, ALENEX/ANALCO 2004*, pp. 39–48 (2004)
101. Lengauer, T.: Black-white pebbles and graph separation. *Acta Informatica* 16, 465–475 (1981)
102. Lipton, R.J., Tarjan, R.E.: A separator theorem for planar graphs. *SIAM Journal on Applied Mathematics* 36, 177–189 (1979)
103. Lipton, R.J., Tarjan, R.E.: Applications of a planar separator theorem. *SIAM Journal on Computing* 9, 615–627 (1980)
104. Lokshtanov, D., Marx, D., Saurabh, S.: Known algorithms on graphs on bounded treewidth are probably optimal. In: Randall, D. (ed.) *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011*, pp. 777–789 (2011)
105. Makowsky, J.A.: Coloured Tutte polynomials and Kauffman brackets for graphs of bounded tree width. *Discrete Applied Mathematics* 145, 276–290 (2004)
106. Mata-Montero, E.: Resilience of partial k -tree networks with edge and node failures. *Networks* 21, 321–344 (1991)
107. Matoušek, J., Thomas, R.: Algorithms for finding tree-decompositions of graphs. *Journal of Algorithms* 12, 1–22 (1991)
108. McDiarmid, C., Reed, B.: Channel assignment on graphs of bounded treewidth. *Discrete Mathematics* 273, 183–192 (2003)
109. Reed, B.: Finding approximate separators and computing tree-width quickly. In: *Proceedings of the 24th Annual Symposium on Theory of Computing, STOC 1992*, pp. 221–228. ACM Press, New York (1992)
110. Robertson, N., Seymour, P.D.: Graph minors. I. Excluding a forest. *Journal of Combinatorial Theory, Series B* 35, 39–61 (1983)
111. Robertson, N., Seymour, P.D.: Graph minors. III. Planar tree-width. *Journal of Combinatorial Theory, Series B* 36, 49–64 (1984)
112. Robertson, N., Seymour, P.D.: Graph minors — a survey. In: Anderson, I. (ed.) *Surveys in Combinatorics*, pp. 153–171. Cambridge Univ. Press (1985)
113. Robertson, N., Seymour, P.D.: Graph minors. II. Algorithmic aspects of tree-width. *Journal of Algorithms* 7, 309–322 (1986)
114. Robertson, N., Seymour, P.D.: Graph minors. V. Excluding a planar graph. *Journal of Combinatorial Theory, Series B* 41, 92–114 (1986)
115. Robertson, N., Seymour, P.D.: Graph minors. VI. Disjoint paths across a disc. *Journal of Combinatorial Theory, Series B* 41, 115–138 (1986)
116. Robertson, N., Seymour, P.D.: Graph minors. VII. Disjoint paths on a surface. *Journal of Combinatorial Theory, Series B* 45, 212–254 (1988)

117. Robertson, N., Seymour, P.D.: Graph minors. IV. Tree-width and well-quasi-ordering. *Journal of Combinatorial Theory, Series B* 48, 227–254 (1990)
118. Robertson, N., Seymour, P.D.: Graph minors. IX. Disjoint crossed paths. *Journal of Combinatorial Theory, Series B* 49, 40–77 (1990)
119. Robertson, N., Seymour, P.D.: Graph minors. VIII. A Kuratowski theorem for general surfaces. *Journal of Combinatorial Theory, Series B* 48, 255–288 (1990)
120. Robertson, N., Seymour, P.D.: Graph minors. X. Obstructions to tree-decomposition. *Journal of Combinatorial Theory, Series B* 52, 153–190 (1991)
121. Robertson, N., Seymour, P.D.: Graph minors. XXII. Irrelevant vertices in linkage problems (1992) (manuscript)
122. Robertson, N., Seymour, P.D.: Graph minors. XI. Distance on a surface. *Journal of Combinatorial Theory, Series B* 60, 72–106 (1994)
123. Robertson, N., Seymour, P.D.: Graph minors. XII. Excluding a non-planar graph. *Journal of Combinatorial Theory, Series B* 64, 240–272 (1995)
124. Robertson, N., Seymour, P.D.: Graph minors. XIII. The disjoint paths problem. *Journal of Combinatorial Theory, Series B* 63, 65–110 (1995)
125. Robertson, N., Seymour, P.D.: Graph minors. XIV. Extending an embedding. *Journal of Combinatorial Theory, Series B* 65, 23–50 (1995)
126. Robertson, N., Seymour, P.D.: Graph minors. XV. Giant steps. *Journal of Combinatorial Theory, Series B* 68, 112–148 (1996)
127. Robertson, N., Seymour, P.D.: Graph minors. XVII. Taming a vortex. *Journal of Combinatorial Theory, Series B* 77, 162–210 (1999)
128. Robertson, N., Seymour, P.D.: Graph minors. XVI. Excluding a non-planar graph. *Journal of Combinatorial Theory, Series B* 89, 43–76 (2003)
129. Robertson, N., Seymour, P.D.: Graph minors. XVIII. Tree-decompositions and well-quasi ordering. *Journal of Combinatorial Theory, Series B* 89, 77–108 (2003)
130. Robertson, N., Seymour, P.D.: Graph minors. XIX. Well-quasi-ordering on a surface. *Journal of Combinatorial Theory, Series B* 90, 325–385 (2004)
131. Robertson, N., Seymour, P.D.: Graph minors. XX. Wagner’s conjecture. *Journal of Combinatorial Theory, Series B* 92, 325–357 (2004)
132. Robertson, N., Seymour, P.D.: Graph minors. XXI. Graphs with unique linkages. *Journal of Combinatorial Theory, Series B* 99, 583–616 (2009)
133. Robertson, N., Seymour, P.D.: Graph minors XXIII. Nash-Williams’ immersion conjecture. *Journal of Combinatorial Theory, Series B* 100, 181–205 (2010)
134. Robertson, N., Seymour, P.D., Thomas, R.: Quickly excluding a planar graph. *Journal of Combinatorial Theory, Series B* 62, 323–348 (1994)
135. Sanders, D.P.: On linear recognition of tree-width at most four. *SIAM Journal on Discrete Mathematics* 9(1), 101–117 (1996)
136. Sau, I., Thilikos, D.M.: Subexponential parameterized algorithms for degree-constrained subgraph problems on planar graphs. *Journal of Discrete Algorithms* 8, 330–338 (2010)
137. Seymour, P.D., Thomas, R.: Call routing and the ratcatcher. *Combinatorica* 14(2), 217–241 (1994)
138. Telle, J.A., Proskurowski, A.: Algorithms for vertex partitioning problems on partial k -trees. *SIAM Journal on Discrete Mathematics* 10, 529–550 (1997)
139. Thilikos, D.M., Serna, M.J., Bodlaender, H.L.: Cutwidth I: A linear time fixed parameter algorithm. *Journal of Algorithms* 56, 1–24 (2005)
140. Thilikos, D.M., Serna, M.J., Bodlaender, H.L.: Cutwidth II: Algorithms for partial w -trees of bounded degree. *Journal of Algorithms* 56, 25–49 (2005)
141. van Rooij, J.M.M.: Exact exponential-time algorithms for domination problems in graphs. PhD thesis, Department of Computer Science, Utrecht University (2011)

142. van Rooij, J.M.M., Bodlaender, H.L., Rossmanith, P.: Dynamic Programming on Tree Decompositions Using Generalised Fast Subset Convolution. In: Fiat, A., Sanders, P. (eds.) ESA 2009. LNCS, vol. 5757, pp. 566–577. Springer, Heidelberg (2009)
143. Wimer, T.V.: Linear Algorithms on k -Terminal Graphs. PhD thesis, Dept. of Computer Science, Clemson University (1987)
144. Wimer, T.V., Hedetniemi, S.T., Laskar, R.: A methodology for constructing linear graph algorithms. *Congressus Numerantium* 50, 43–60 (1985)
145. Zhou, X., Fuse, K., Nishizeki, T.: A linear algorithm for finding $[g, f]$ -colorings of partial k -trees. *Algorithmica* 27, 227–243 (2000)

Graph Minors and Parameterized Algorithm Design*

Dimitrios M. Thilikos

Department of Mathematics, National and Kapodistrian University of Athens,
Panepistimioupolis, GR-15784, Athens, Greece
sedthilk@math.uoa.gr

Abstract. The Graph Minors Theory, developed by Robertson and Seymour, has been one of the most influential mathematical theories in parameterized algorithm design. We present some of the basic algorithmic techniques and methods that emerged from this theory. We discuss its direct meta-algorithmic consequences, we present the algorithmic applications of core theorems such as the grid-exclusion theorem, and we give a brief description of the irrelevant vertex technique.

Keywords: graph minors, parameterized algorithms, treewidth, bidimensionality, irrelevant vertex technique, linkages.

1 Introduction

Graph Minors Theory (GMT) was developed by Robertson and Seymour in a series of 23 papers, between 1984 and 2009. Among them, the second paper of the series was published in the *Journal of Algorithms* while all the rest were published in the *Journal of Combinatorial Theory Series B*. The main theoretical achievement of this project was the proof of Wagner’s conjecture, now known as the *Robertson & Seymour Theorem*, stating that graphs are well-quasi-ordered under the minor containment relation. Besides its purely mathematical importance, GMT induced a series of powerful algorithmic results and techniques that had a deep influence on theoretical computer science. More particularly, GMT has been one of the most powerful “mathematical engines” in the theory and design of parameterized algorithms. In particular, a considerable part of the basic techniques in parameterized algorithm design is directly or indirectly linked to results from GMT. Moreover, GMT offered the theoretical base for the understanding and resolution of some of the most prominent graph-algorithmic problems in parameterized complexity. In what follows, we give a brief presentation of the main results and techniques in this area.

* This research has been co-financed by the European Union (European Social Fund – ESF) and Greek national funds through the Operational Program “Education and Lifelong Learning” of the National Strategic Reference Framework (NSRF) - Research Funding Program: “*Thalis. Investing in knowledge society through the European Social Fund*”.

Our presentation is organized as follows. In Section 2 we give the definitions of some basic combinatorial and algorithmic concepts. In Section 3.1 we present the main algorithmic consequences of the GMT, mainly from the parameterized complexity viewpoint. Section 4 is devoted to the celebrated grid-exclusion theorem and its applications to bidimensionality theory. Finally, Section 5, attempts a short presentation of the irrelevant vertex technique and its applications.

2 Basic Definitions

All graphs we consider are finite, undirected and simple, i.e., they do not have multiple edges or loops. Given a graph G we denote by $V(G)$ and $E(G)$ its vertex and edge set respectively. The *size* (reps. *magnetite*) of a graph G is the number of its vertices (reps. edges) and is denoted by $n(G)$ (reps. $m(G)$), i.e., $n(G) = |V(G)|$ ($m(G) = |E(G)|$). We denote by $G \setminus v$ the graph obtained by removing v (along with its incident edges) from G . The *neighborhood* of a vertex $v \in V(G)$, denoted by $N_G(v)$, is the set of edges in G that are adjacent to v . The *degree* of a vertex $v \in V(G)$ is the cardinality of its neighborhood in G . We denote by K_r the complete graph on r vertices and by $K_{r,q}$ the complete bipartite graph with r vertices in its one part and q in the other. Finally, we denote by G_k the $(k \times k)$ -grid, i.e., the Cartesian product of two paths of length $k - 1$ (see Figure 1).

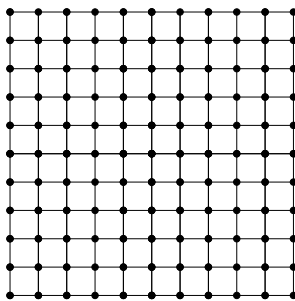


Fig. 1. The $(11, 11)$ -grid G_{11}

2.1 Relations on Graphs and Obstructions

We say that a graph H is a *subgraph* of a graph G if H can be obtained by G by removing edges or vertices. The contraction of an edge $e = \{x, y\}$ from G is the removal from G of all edges incident to x or y and the insertion of a new vertex v_e that is made adjacent to all the vertices of $(N_G(x) \setminus \{y\}) \cup (N_G(y) \setminus \{x\})$. Given two graphs H and G , we say that H is a *contraction* of G , denoted by $H \leq_c G$, if H can be obtained from G by a (possibly empty) series of edge contractions.

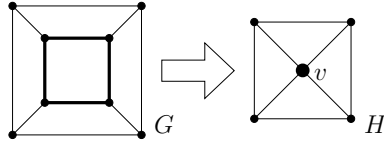


Fig. 2. The graph H is the result of the contractions of the bold edges in G to the vertex v

H is a *minor* of G if H is a contraction of some subgraph of G . A graph H is a *topological minor* of G (denoted by $H \leq_t G$) if G contains as a subgraph some subdivision of H (a *subdivision* of a graph H is any graph obtained by replacing some of its edges by paths between the same endpoints). Given a partial ordering relation \leq on graphs, we say that a graph class \mathcal{G} is *closed under* \leq if for every $G \in \mathcal{G}$, $H \leq G$ implies that $H \in \mathcal{G}$. Let \mathcal{G} be a graph class that is closed under the minor relation. An \leq -*anti-chain* is a set of graphs that are pairwise non-comparable with respect to \leq . For example the set of graphs $\mathcal{A} = \{K_{2,r} \mid r \geq 2\}$ is a \leq_c -antichain but not an \leq_m -antichain or a \leq_t -antichain.

We define the \leq -*obstruction set* of a graph class \mathcal{G} , denoted by $\mathbf{obs}_{\leq}(\mathcal{G})$, as the set of all \leq -minimal graphs that *do not* belong to \mathcal{G} . Clearly, by definition, the \leq -obstruction set of a graph class is an \leq -anti-chain. Obstruction sets can be seen as alternative characterizations of graph classes and, in many cases, reveal a good deal from their structural characteristics. For example, it easy to verify that $\mathbf{obs}_{\leq_m}(\mathcal{T}) = \mathbf{obs}_{\leq_t}(\mathcal{T}) = \{K_3\}$, $\mathbf{obs}_{\leq_m}(\mathcal{O}) = \mathbf{obs}_{\leq_t}(\mathcal{O}) = \{K_4, K_{2,3}\}$, where \mathcal{T} and \mathcal{O} are the classes of all acyclic and all outerplanar graphs respectively, and $\mathbf{obs}_{\leq_c}(\mathcal{O}^*) = \{K_4, K_{2,3}, K_{2,3}^+\}$ where \mathcal{O}^* are the connected outerplanar graphs and $K_{2,3}^+$ is the graph obtained from K_5 by removing a triangle. The most classic theorems on obstruction characterization of graph classes are the Kuratowski-Pontryagin’s theorem [87] and Wagner’s theorem [121], stating that $\mathbf{obs}_{\leq_m}(\mathcal{P}) = \{K_5, K_{3,3}\}$ and $\mathbf{obs}_{\leq_t}(\mathcal{P}) = \{K_5, K_{3,3}\}$ respectively, where \mathcal{P} is the class of all planar graphs.

2.2 Parameterized Problems and Algorithms

The idea of problem parameterization is to treat algorithmic problems as parameterized entities and to evaluate the complexity of the corresponding algorithms by considering the way parameters appear in their running times. As here we deal with problems on graphs, we adapt the classic definitions of parameterized complexity to the case where problem inputs represent graphs.

Formally, a *parameterized problem on graphs* is a subset Π of $\Sigma^* \times \mathbb{N}$, where Σ is some alphabet and, in each $(I, k) \in \Sigma^* \times \mathbb{N}$, I encodes a combinatorial structure related to one, or more, graphs. For this, we agree that n (resp. m) is the maximum size (resp. magnitude) of the graphs encoded in I and we insist that $|I, k| = O(m)$. We call I the *main part of the input* and we say that k is the *parameter* of the problem. Two instances $(I, k), (I', k') \in \Sigma^* \times \mathbb{N}$ are *equivalent with respect to Π* if $(I, k) \in \Pi \iff (I', k') \in \Pi$.

We say that Π is *fixed parameter tractable* if there exists a function¹ $f : \mathbb{N} \rightarrow \mathbb{N}$ and an algorithm deciding whether $(I, k) \in \Pi$ in $O(f(k) \cdot n^c)$ steps, where c is a constant not depending on the parameter k of the problem. We call such an algorithm *FPT-algorithm* or, to express concretely the choice of f and c , we say that $\Pi \in O(f(k) \cdot n^c)$ -FPT. A parameterized problem on graphs belongs to the parameterized class FPT if it can be solved by an FPT-algorithm. In fact, not all parameterized problems belong to the class FPT. There is a hierarchy of parameterized complexity classes, namely

$$\text{FPT} \subseteq \text{W}[1] \subseteq \text{W}[2] \subseteq \text{W}[3] \subseteq \dots \subseteq \text{W}[\text{SAT}] \subseteq \text{W}[\text{P}] \subseteq \text{XP},$$

and appropriate *parameter-preserving* reductions such that, when a problem is hard for some of them (other than FPT), it is not expected to have an FPT-algorithm (all inclusions in this hierarchy are believed to be strict). See the monographs [39, 44, 97] for more details on parameterized complexity.

Time bounds for parameterized algorithms have two parts. The term $f(k)$ is called *parameter dependence* and, is typically a super-polynomial function. On the other hand, the term n^c is a polynomial function and we call it *polynomial part*. In most of the problems that we examine here, I will encode a simple graph. To simplify notation, we frequently write “ $O_k(n^c)$ ” instead of “ $f(k) \cdot (n^c)$ ” for some recursive function $f : \mathbb{N} \rightarrow \mathbb{N}$ and, in this case, we refer to the function f hidden in the O_k notation as the *parameter dependence*.

3 Algorithmic Consequences of the GMT

3.1 Well-Quasi-Ordering

The main combinatorial result of the GMT fits in the more general framework of the theory of Well-Quasi-Orderings, first developed by Graham Higman² under the name “finite basis property” [65]. Given a set \mathcal{X} and a partial ordering \leq on \mathcal{X} we say that \mathcal{X} is *well-quasi-ordered under \leq* if none of its subsets is an infinite \leq -antichain.

Theorem 1 (Robertson & Seymour Theorem [110]). *The set of all graphs is well-quasi-ordered under minors.*

In other words, Theorem 1 says that If \mathcal{G} is an infinite set of graphs then there exist two graphs $H, G \in \mathcal{G}$ such that H is a minor of G . The proof of theorem 1 was concluded in paper XX of the Graph Minors Series. Before its proof, the statement of Theorem 1 was known as Wagner’s conjecture. However, as mentioned by Diestel in [32], Wagner said that he had never made such a conjecture. A similar conjecture, on the well-quasi-ordering of trees under the topological minor

¹ Notice that in the definition of FPT f is not necessarily a recursive function.

² As mentioned in [86], the same theory was also developed in some unpublished manuscript of Erdős and Rado, while its first hints can be traced back to B. H. Neumann [96]

relation was made by Vázsonyi and was proved in 1960 independently³ by Joseph Kruskal and S. Tarkowski [91]. Interesting results on the meta-mathematics of Kruskal’s tree theorem as well as Roberson & Seymour’s theorem can be found in [55] and [54] respectively.

Consider the following parameterized problem:

H-MINOR CHECKING
Instance: Two graphs *G* and *H*.
Parameter: $k = |V(H)|$.
Question: Is *H* a minor of *G*?

The main algorithmic contribution of the GMT is the following result.

Theorem 2 (Robertson and Seymour [108]). *One can construct an algorithm that, given a n -vertex graph G and a k -vertex graph H , checks whether H is a minor of a graph G in $O_k(n^3)$ steps. In other words, H -MINOR CHECKING $\in O_k(n^3)$ -FPT.*

Actually, Robertson and Seymour in [108] describe an $O_k(n^3)$ -step algorithm that solves a generalization of the H -MINOR CHECKING and another celebrated problem, namely the k -DISJOINT PATHS problem. In Section 5, we give a rough description of the main ingredients of the algorithm in Theorem 2 especially for the k -DISJOINT PATHS problem. Recently, this running time was improved to a quadratic one for the k -DISJOINT PATHS problem in [74].

The good news about Theorem 2 is that it is constructive (contrary to Theorem 1) and there is a recursive function hidden in the O_k notation. The bad news is that, according to the algorithm in [108] and the proof of its correctness in [111] and [107], the values of this function are *immense*⁴, even for small values of k . David Johnson mentioned in [67]:

“for any instance $G = (V, E)$ that one could fit into the known universe, one would easily prefer $|V|^{70}$ to even constant time, if that constant had to be one of Robertson and Seymour’s”.

Moreover, in [67], David Johnson estimates that just one constant in the parameter dependence of Theorem 2 is roughly

$$2^{12^{2^{2^{2^{12^{\uparrow e(r)}}}}}}$$

where $2^{\uparrow r}$ denotes a tower $2^{2^{\dots}}$ involving r 2’s. Clearly, such type of constants may create reasonable doubts to computer scientists on whether such an algorithm may be considered to be an “algorithm” of some practical meaning. In fact, to investigate until which point these constants can be improved is an open and challenging problem in parameterized complexity and algorithms (see e.g. [3]).

³ A shorter and quite elegant proof of Vázsonyi’s conjecture was given by Nash-Williams in 1963 [95].

⁴ Perhaps the word “immense” is somehow moderate here. Instead, Fellows and Langston used the expression “mind-boggling” in [43].

3.2 Minor-Closed Graph Parameters

A *parameter on graphs* (or a *graph parameter*) is any function that maps graphs to integers and with the property that it is invariant under graph isomorphism. Let \leq be a relation on graphs. We say that a graph parameter \mathbf{p} is *closed under \leq* (or, simply, *\leq -closed*) if for every two graph H and G , $H \leq G$ implies that $\mathbf{p}(H) \leq \mathbf{p}(G)$. We define the *\leq -obstruction family* of \mathbf{p} as the parameterized graph class

$$\mathcal{O}_{\mathbf{p},k}^{\leq} = \mathbf{obs}_{\leq}(\{G \mid \mathbf{p}(G) \leq k\}).$$

Consider the following parameterized meta-problem.

k -PARAMETER CHECKING FOR \mathbf{p}
Instance: a graph G and an integer $k \geq 0$.
Parameter: k
Question: $\mathbf{p}(G) \leq k$?

Theorems 1 and 2 together have the following dramatic consequence.

Theorem 3. *For every parameter \mathbf{p} that is closed under minors there exists an algorithm that solves the problem k -PARAMETER CHECKING FOR \mathbf{p} in $f(k) \cdot n^3$ steps for some function f .*

Proof. Recall that, by definition, no two graphs in $\mathcal{O}_{\mathbf{p},k}^{\leq m}$ can be comparable graphs under the minor relation. It follows, from Theorem 1, that $\mathcal{O}_{\mathbf{p},k}^{\leq m}$ is a finite set. Let $g(k) = |\mathcal{O}_{\mathbf{p},k}^{\leq m}|$. As \mathbf{p} is closed under minors, it holds that

$$\mathbf{p}(G) \leq k \iff \forall H \in \mathcal{O}_{\mathbf{p},k}^{\leq m} \quad H \not\leq_m G.$$

Therefore, to check whether $\mathbf{p}(G) \leq k$ it is enough to apply $g(k)$ times the $\mathcal{O}_k(n^3)$ step algorithm of Theorem 2 and check whether some member of $\mathcal{O}_{\mathbf{p},k}^{\leq m}$ is contained as a minor in G .

Theorem 3 had a great impact in parameterized complexity as it implied a massive classification of problems in the class FPT. In that sense, Theorem 3 is an algorithmic meta-theorem because it provides a generic condition (minor-closedness) for a parameterized problem that automatically implies the existence of an FPT-algorithm for it. Unfortunately, the proof of Theorem 1 does not provide any general “meta-algorithm” to compute the set $\mathcal{O}_{\mathbf{p},k}^{\leq m}$ and, that way, construct the claimed algorithm for each \mathbf{p} . In fact, due to the meta-mathematics of Theorem 1 [54], such an meta-algorithm does not exist. As observed in [43], there is no algorithm that, given a Turing machine accepting precisely the graphs of a minor-closed graph class \mathcal{F} , outputs $\mathbf{obs}_{\leq m}(\mathcal{F})$ (see also [119]). However, Theorem 3 gave important (mathematical) energy to Parameterized Algorithms as it acted as an “encouraging factor”. The knowledge that an algorithm exists for a specific problem, induces the challenge to construct one and, in a sense, provides the courage to try to accomplish such a task.

In order to cope with the inherent non-constructivity of Theorem 3 one may study specific parameters where the computation of the set $\mathcal{O}_{\mathbf{p},k}^{\leq m}$ (or, at least, of some upper bound to the function $g(k)$) in the proof of Theorem 3 is possible. However, this is not an easy task, even for simple parameters. According to [34], if the problem of checking whether $\mathbf{p}(G) \leq k$ is NP-complete, then $|\mathcal{O}_{\mathbf{p},k}^{\leq m}|$ is a super-polynomial function of k , unless the polynomial hierarchy collapses to Σ_3^P . Characterizations of $\mathbf{p}(G) \leq k$ (yielding better lower bounds for $|\mathcal{O}_{\mathbf{p},k}^{\leq m}|$) have been provided for several parameters [10, 20, 40, 58, 84, 98, 99, 115, 117, 118]. However, to our knowledge, there is not yet a natural parameter \mathbf{p} for which a complete characterization of $\mathcal{O}_{\mathbf{p},k}^{\leq m}$ is known. A more promising strategy towards detecting constructive fragments of Theorem 3, is to detect parameters – or families of parameters – where $\mathcal{O}_{\mathbf{p},k}^{\leq m}$ is recursive. For this one may either prove upper bounds for $|\mathcal{O}_{\mathbf{p},k}^{\leq m}|$, as done in [57] for the case of branchwidth⁵, or provide partial characterizations of $\mathcal{O}_{\mathbf{p},k}^{\leq m}$, as done in [2, 19, 21, 89, 94], that permit its recursive computation.

At this point, we should mention that all theorems of this section have their counterparts in another partial relation on graphs, the one of *immersion*. The *lift* of two incident edges is the operation of removing two edges $e_1 = \{x, y\}$ and $e_2 = \{x, z\}$ (incident to a common vertex x) and adding the edge $\{y, z\}$. We say that a graph H can be *immersed* in a graph G , denoted by $H \leq_{im} G$, if H can be obtained from a subgraph of G by a (possibly empty) sequence of edge lifts. According to the last paper of the Graph Minor series [112], graphs are well-quasi-ordered under immersions, i.e., Theorem 1 holds also if we replace minors by immersions. Therefore, in order to prove a counterpart of Theorem 3 for the case of immersion-closed parameters, we need an algorithm that given an n -vertex graph G and a k -vertex graph H , checks where $H \leq_{im} G$ in $O_k \cdot (n(G))^{3k}$ steps. Recently, a construction of such an algorithm was given in [61]. This makes it possible to derive the following meta-algorithmic result.

Theorem 4. *If \mathbf{p} is a parameter that is closed under immersions, then there exists an algorithm that solves the problem k -PARAMETER CHECKING FOR \mathbf{p} in $f'(k) \cdot n^3$ steps for some function f' .*

In fact, the main result of [61] proves the FPT membership of topological minor testing, i.e., given two graphs H and G , check whether $H \leq_t G$ (the parameter is the size of H). This means that there is a counterpart of Theorem 2 for the topological minor relation as well. This might create some hope that Theorem 3 holds for topological minors as well. Unfortunately, this requires an analogue of the combinatorial Theorem 1 which does not exist as it is possible to construct an infinite class of graphs that are pairwise non-comparable with respect to the topological minor relation: just take all cycles with their edges duplicated.

⁵ Branchwidth was introduced in the paper X of the Graph Minor Series [106] and, from that point and then, was used as an alternative for treewidth (defined formally in Section 4). Treewidth and branchwidth can be seen as twin parameters, as the one is a constant factor approximation of the other.

An other argument for the non-existence of analogues of Theorems 1 and 3 for topological minors is given by the TOPOLOGICAL BANDWIDTH problem asking whether the topological bandwidth of a graph is at most k . The topological bandwidth of a graph G is denoted by $\mathbf{tbw}(G)$ and is defined as

$$\mathbf{tbw}(G) = \min\{k \mid \exists q \geq 1: G \leq_t P_q^k\}$$

(P_q^k is obtained by a path P_q of length q if we make adjacent any two vertices of distance $\leq k$ in P_q). It is easy to observe that \mathbf{tbw} is closed under topological minors. In [42] it is mentioned that TOPOLOGICAL BANDWIDTH is $W[t]$ -hard for all $t \geq 1$ – the proof is a modification of the proof for the case of BANDWIDTH in [15]. This implies that, under reasonable assumptions in parameterized complexity theory, the anti-chain corresponding to the \leq_t -obstruction family $\mathcal{O}_{\mathbf{tbw},k}^{\leq t}$ is infinite for an infinite set of values of k .

4 Grid-Exclusion and Bidimensionality

4.1 Treewidth

Treewidth has been one of the main contributions of GMT to algorithmic graph theory. While, as a concept, its indices can be traced back to the work of Gavril in [56], its formal birth as a graph parameter occurred in the second paper of the Graph Minors series [104]. Currently, there are at least six equivalent definitions of tree-width. We present the original one from [104].

A *tree decomposition* of a graph G is a pair (\mathcal{X}, T) where T is a tree and $\mathcal{X} = \{X_i \mid i \in V(T)\}$ is a collection of subsets of $V(G)$ such that:

1. $\bigcup_{i \in V(T)} X_i = V(G)$;
2. for each edge $\{x, y\} \in E(G)$, $\{x, y\} \subseteq X_i$ for some $i \in V(T)$, and
3. for each $x \in V(G)$ the set $\{i \mid x \in X_i\}$ induces a connected subtree of T .

The *width* of a tree decomposition $(\{X_i \mid i \in V(T)\}, T)$ is $\max_{i \in V(T)} \{|X_i| - 1\}$. The *treewidth* of a graph G , denoted by $\mathbf{tw}(G)$, is the minimum width over all tree decompositions of G .

If, in the above definitions, we restrict the tree T to be a path then we define the notions of *path decomposition* and *pathwidth*. We write $\mathbf{pw}(G)$ to denote the pathwidth of a graph G . Pathwidth was defined earlier than treewidth in the first paper is the Graph Minors Series [102].

Treewidth can intuitively be seen as a measure of the topological resemblance of a graph to a tree or, alternatively, as a measure of the “global connectivity” of a graph. Similarly, pathwidth can be seen as a measure of the topological resemblance of a graph to a path.

Counting Monadic Second Order Logic (CMSOL) is a logic on graphs⁶ where the domain is the set of vertices and edges, there are predicates for vertex-vertex

⁶ We should stress that CMSOL is not only a logic on graphs but also on more general combinatorial objects called *strucures*.

adjacency and edge-vertex incidence, there is quantification over edges, vertices, edge sets and vertex sets, and there is a predicate $Card_{r,p}(S)$ which expresses whether the size of a set S is r modulo p .

The importance of treewidth for algorithmic graph theory is illustrated by the celebrated Courcelle’s theorem stating that if Π_k is a parameterized property of graphs expressible by a CMSOL formula ϕ_k , then there is an algorithm that, given as input a graph G , can check whether G satisfies property Π_k (i.e., whether $G \in \Pi_k$) in $O_{|\phi_k|+\text{tw}(G)}(n)$ steps. Moreover, there exists a meta-algorithm that, given ϕ_k , outputs such an algorithm. A proof of Courcelle’s theorem can be found in [39, Chapter 6.5] and [44, Chapter 10] and similar results appeared by Arnborg, Lagergren, and Seese in [8] and Borie, Parker, and Tovey in [17]. An alternative game-theoretic proof has appeared recently in [81, 82].

Courcelle’s theorem had a deep influence in parameterized algorithms as it automatically yields FPT-algorithms for a wide family of problems, provided that the treewidth of their instances is bounded by a function of the parameter k . The natural challenge is whether and when a parameterized problem can be reduced to its bounded treewidth variant. For this, an important step is to detect what kind of combinatorial structures are contained in a graph with big treewidth. The most prominent structure of this type is the grid G_k . Let $\mathbf{gm}(G)$ be the maximum k for which G contains G_k as a minor. A valuable theoretical tool in this direction was given by the following result of the GMT.

Theorem 5 [105]. *There exists a recursive function $f : \mathbb{N} \rightarrow \mathbb{N}$ such that $\text{tw}(G) \leq f(\mathbf{gm}(G))$.*

While the above result appeared in the fifth paper of the series, a preliminary variant of it, where G is planar, appeared earlier in [103]. As every graph containing G_k as a minor has treewidth at least k , Theorem 5 implies that tw and \mathbf{gm} are parametrically equivalent: a bound to the one of them implies a bound to the other. The initial estimation of the parameter dependence in Theorem 5 was huge. However, a better one appeared in [113] where it was proven that $\text{tw}(G) = 20^{2 \cdot (\mathbf{gm}(G))^5}$. An alternative, and relatively simpler, proof of Theorem 5 was given in [33]. To see the use of Theorem 5 in parameterized algorithm design, consider a parameter \mathbf{p} that satisfies the following properties:

- i. \mathbf{p} is closed under taking of minors.
- ii. there exists a recursive function $t : \mathbb{N} \rightarrow \mathbb{N}$ such that $\mathbf{p}(G_{t(k)}) > k$ for every non-negative integer k .
- iii. One can construct an algorithm that, given a tree-decomposition of G of width at most q and an integer k , checks whether $\mathbf{p}(G) \leq k$ in $l(k, q) \cdot n^{O(1)}$ steps for some recursive function $l : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$.

Clearly, the first two conditions are easy to check for most instantiations of \mathbf{p} . Moreover, the third one follows directly from Courcelle’s theorem if for each k , $\Pi_k = \{G \mid \mathbf{p}(G) \leq k\}$ is expressible by a CMSOL formula ϕ_k . There are many examples of such parameters. Typical examples are the vertex cover of a graph, i.e., the minimum number of vertices that meets all vertices of G and the

feedback vertex set of a graph, i.e., the minimum number of vertices meeting all cycles of G . A direct consequence of Theorem 5 is the following (constructive) special case of Theorem 3:

Lemma 1. *Let \mathbf{p} be a parameter satisfying conditions **i–iii** above for some t and l . Then it is possible to construct an algorithm that, given as input a graph G and an integer k , checks whether $\mathbf{p}(G) \leq k$ in $(2^{O(f(t(k)))} + l(k, 4 \cdot f(t(k)))) \cdot n^{O(1)}$ steps where f is the function in Theorem 5.*

Proof. The algorithm in Lemma 1 works as follows: First of all, it uses an FPT-approximation algorithm for treewidth, i.e., an algorithm that given a graph G and an integer q , either outputs a tree decomposition of G of width at most $\alpha \cdot q$ or reports that $\mathbf{tw}(G) > q$ in $z(q) \cdot n^\beta$ steps. Various algorithms of this type have been proposed in [7, 14, 88, 100, 108] for different trade-offs between z , α , and β . Among them, we pick the one from [7] where $z(q) = 2^{4 \cdot 38 \cdot q}$, $\alpha = 4$, and $\beta = 2$. We run this algorithm for G and $q = f(t(k))$. If it outputs a tree decomposition of width $\leq 4 \cdot f(t(k))$ then we use the algorithm of Property **iii** and solve the problem in $l(k, 4 \cdot f(t(k))) \cdot n^{O(1)}$ steps. If the algorithm reports that the treewidth of G is more than $f(t(k))$, then from Theorem 5, G contains $G_{t(k)}$ as a minor. In such a case, the algorithm directly outputs a negative answer as, from Properties **i** and **ii**, $\mathbf{p}(G) \geq \mathbf{p}(G_{t(k)}) > k$.

The idea of the above proof is also known as the *Win/win approach*: we either have an answer to the problems directly because the treewidth is big enough or we solve the problems using dynamic programming on a tree-decomposition of bounded width.

Clearly, the running time of the algorithm in Lemma 1 depends on the functions f , t , and l . In what follows, we comment on the current bounds on each one of them.

- l*: As we have already mentioned, the (constructive) existence of l , follows from Courcelle’s theorem for the wide family of problems that are expressible in CMSOL. However, the bounds on l , derived from the proof, are huge and this may dismiss any hope for a good parameter dependence (see [53]). However, for many problems it is possible to directly apply dynamic programming on the tree decomposition and derive moderate bounds on l such as $l(k, q) = 2^{O(q^2)} \cdot k^{O(1)}$, or $l(k, q) = 2^{O(q \log q)} \cdot k^{O(1)}$ or, even better, $l(k, q) = 2^{O(q)} \cdot k^{O(1)}$. Clearly, time bounds of the third type are more attractive. For this reason, we say that a parameter \mathbf{p} is *single exponentially solvable with respect to treewidth* if there exists an algorithm that, given G and k , checks whether $\mathbf{p}(G) \leq k$ in $2^{O(\mathbf{tw}(G))} \cdot n^{O(1)}$ steps. There is a quite extended bibliography on how to do fast dynamic programming on graphs of bounded treewidth; as a sample of this, we just mention [5, 6, 9, 11, 13, 16, 22, 35, 35, 36, 37, 37, 38, 114, 120, 120].
- t*: Bounds are much better for the function t . For most natural graph parameters, it holds that $t(k) = O(k)$ while for some of them, including \mathbf{tw} and \mathbf{pw} , it holds that $t(k) = \Theta(k)$. However, there is a wide family of parameters

where $t(k) = O(\sqrt{k})$. This intuitively says that a certificate for the value of such a parameter spreads “bidimensionally” inside a $(k \times k)$ -grid. For instance any vertex cover of G_k should have size at least $k \cdot \lfloor \frac{k}{2} \rfloor = \Omega(k^2)$ as the vertices of such a set should cover edges all over the “area” of the grid. Similarly, a feedback vertex set of G_k should have size at least $(\lfloor \frac{k}{2} \rfloor)^2 = \Omega(k^2)$ as the vertices of such a set should cover all $(\lfloor \frac{k}{2} \rfloor)^2$ members of a packing of “squares” in G_k .

If such a parameter is also closed under taking of minors then we call it *minor bidimensional*.

f: To improve the function f , i.e., the parameter dependence in Theorem 5, is an important challenge as, even for the parameters with most moderate instantiations of l and t , k -PARAMETER CHECKING FOR \mathbf{p} could be only classified in $2^{2^{k^{O(1)}}} \cdot n^{O(1)}$ -FPT. Robertson, Seymour, and Thomas conjectured in [113] that f can be a polynomial function. This would directly imply that k -PARAMETER CHECKING FOR \mathbf{p} belongs to $2^{k^{O(1)}} \cdot n^{O(1)}$ -FPT for a wide family of parameters (see [30] for more discussions and conjectures on this issue).

Another interesting problem is to lower bound the contribution of f in Theorem 5. As mentioned by Robertson, Seymour, and Thomas in [113] there are graphs excluding G_k as a minor that have treewidth $\Omega(k^2 \cdot \log k)$. To see this, one may use the result in [23] (see also [41, 122]) to construct an $O(1)$ -regular Ramanujan graph G on n vertices that has girth $\Omega(\log n)$. One can easily verify that $\mathbf{gm}(G) = O(\frac{\sqrt{n}}{\log n})$. The claimed bound follows because Ramanujan graphs are expanders and thus $\mathbf{tw}(G) = \Omega(n)$. It is a challenging question whether any bound better than this one can be proven.

Towards achieving a polynomial dependance between treewidth and the size of an excluded grid, Reed and Wood defined in [101] the notion of a grid-like-minor. A *grid-like-minor* of order k in a graph G is a set of paths in G whose intersection graph is bipartite and contains a K_k as a minor. Clearly, the rows and columns of the $(k \times k)$ -grid are a grid-like-minor of order $k + 1$. In [101] it is proved that every graph with treewidth $\Omega(k^4 \sqrt{\log k})$ contains a grid-like minor of order k . Meta-algorithmic implications of the results in [101], analogous to those of Theorem 1, can be found in [85].

4.2 Bidimensionality

Theorem 5 has several refinements that are important for improving the parameter dependence of the algorithm in Lemma 1. The first variant of Theorem 5 for special graph classes appeared in [113] (proved for the twin parameter of branchwidth) from which it follows that if G is a planar graph, then $\mathbf{tw}(G) \leq 6 \cdot \mathbf{gm}(G)$. Actually, with some more careful application of the results of [113] it can also be proven that $\mathbf{tw}(G) \leq 5 \cdot \mathbf{gm}(G)$, which can be improved further to $\mathbf{tw}(G) \leq \frac{9}{2} \cdot \mathbf{gm}(G)$ using the results of [62]. An analogous upper bound holds also for graphs embedded in surfaces. From the results in [27], it follows that $\mathbf{tw}(G) \leq 6 \cdot (\mathbf{eg}(G) + 1) \cdot \mathbf{gm}(G)$ where $\mathbf{eg}(G)$ is the Euler genus

of G . Also in [30], it was proven that if G is a $K_{3,r}$ -minor free graph, then $\mathbf{tw}(G) \leq 20^{4r} \cdot \mathbf{gm}(G)$. At this point, the natural question is whether this linear dependence holds for every non-trivial minor-closed graph class. This was resolved in [29], where the following theorem has been proved.

Theorem 6. *Let r be a positive integer. If G is a K_r -minor free graph, then $\mathbf{tw}(G) = O_r(\mathbf{gm}(G))$.*

The proof of Theorem 6 is heavily based on GMT. More specifically, it depends on the Structure Theorem of the GMT [109] which implies immense bounds for the parameter dependence of the bound in Theorem 6. The improvement of the parameter dependence of Theorem 6 is an interesting problem and this might be possible without making use of the structural results of [109].

As mentioned in the previous Section, a parameter \mathbf{p} is *minor-bidimensional* if it is closed under taking of minors and for every non-negative integer k it holds that $\mathbf{p}(G_{\lceil \sqrt{k} \rceil}) = \Omega(k)$. A major consequence of Lemma 1, Theorem 6, and the discussion above is the following meta-algorithmic result.

Theorem 7. *Let H be an r -vertex graph and let \mathbf{p} be a graph parameter that is minor-bidimensional and single exponentially solvable with respect to treewidth. Then k -PARAMETER CHECKING FOR \mathbf{p} restricted to H -minor free graphs belongs (constructively) to $2^{O_r(\sqrt{k})} \cdot n^{O(1)}$ -FPT, i.e., one can construct a sub-exponential FPT-algorithm that solves it.*

Notice that the above result is, in a sense, optimal, as, due to the complexity bounds in [18], a $2^{O(\sqrt{k})} \cdot n^{O(1)}$ -step parameterized algorithm is the best we may expect for several bidimensional parameters, even on planar graphs. The meta-algorithmic machinery that we employed above in order to prove Theorem 7 is known as *Bidimensionality Theory* and was introduced for the first time in [27], while some preliminary ideas had already appeared in [4, 52].

Theorem 7 concerns only minor-closed parameters. A typical parameter that does not fit in the framework of minor-bidimensionality is the dominating set number, denoted by $\mathbf{ds}(G)$ and defined as the minimum size of a *dominating set* in G , i.e., a set S of vertices such that every vertex not in S has some neighbor in S .

The dominating set number is not minor-closed as it may increase by removing edges. However this is not the case when we do only contractions. To develop the contraction counterpart of bidimensionality, one has to find a counterpart of Theorem 6 for contractions, i.e., to detect what types of graphs appear as contractions in graphs with big treewidth. This line of research was developed in [26, 31] and concluded in [46]. Before we present the the results in [46], we need first some definitions.

Let Γ_k ($k \geq 2$) be the graph obtained from the $(k \times k)$ -grid by triangulating internal faces of the $(k \times k)$ -grid such that all internal vertices become of degree 6, all non-corner external vertices are of degree 4, and then one corner of degree two is joined by edges with all vertices of the external face (the *corners* are the vertices that in the underlying grid have degree two). Graph Γ_6 is shown

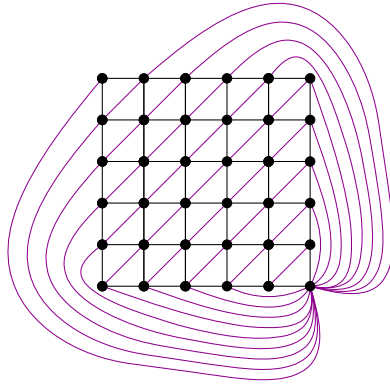


Fig. 3. The graph Γ_6

in Fig. 3. Let also Π_k be the graph obtained from Γ_k by adding a new vertex adjacent to all vertices of Γ_k .

A consequence of the results in [46] is the following.

Theorem 8. *There exists a function $\alpha : \mathbb{N} \rightarrow \mathbb{N}$ such that every connected graph of treewidth at least $\alpha(k)$ contains some of the graphs in $\{K_k, \Gamma_k, \Pi_k\}$ as a contraction.*

Theorem 8 has several refinements. One of them is the following counterpart of Theorem 6.

Theorem 9. *There exists a function $\beta : \mathbb{N} \rightarrow \mathbb{N}$ such that every connected K_r -minor-free graph of treewidth at least $\beta(r) \cdot k^2$ contains either Γ_k or Π_k as a contraction.*

Notice that in the above theorem, the quadratic dependence (on k) is optimal. Indeed, let Z_{k^2} be the graph obtained by adding to G_{k^2} a new vertex adjacent to all the k^2 vertices with both coordinates in the underlying grid divisible by k . Then Z_{k^2} excludes K_6 , G_{k+2} , and Π_{k+2} as contractions and is of treewidth at least k^2 . This means that, in order to have a “linear counterpart” of Theorem 6, we should restrict further the graphs that we exclude. An *apex graph* is a graph that can become planar by the removal of one vertex. It appears that the linear dependence in the bound of Theorem 6 is also possible for contractions when we consider graphs excluding some apex graph as a minor. For this, we define $\mathbf{tgm}(G)$ as the maximum k for which G contains Γ_k as a contraction.

Theorem 10. *Let H be an apex graph with r vertices. If G is a connected H -minor-free graph, then $\mathbf{tw}(G) = O_r(\mathbf{tgm}(G))$.*

We say that a parameter \mathbf{p} is *contraction bidimensional* if it is closed under taking of contractions and if $\mathbf{p}(\Gamma_{\lceil \sqrt{k} \rceil}) = \Omega(k)$ for every non-negative integer

k . Using now Theorem 10 one can derive the following contraction counterpart of Theorem 7.

Theorem 11. *Let H be an r -vertex apex graph and let \mathbf{p} be graph parameter that is contraction-bidimensional and single exponentially solvable with respect to treewidth. Then k -PARAMETER CHECKING FOR \mathbf{p} restricted to H -minor free graphs belongs (constructively) to $2^{O_r(\sqrt{k})} \cdot n^{O(1)}$ -FPT, i.e., one can construct a sub-exponential FPT-algorithm that solves it.*

The algorithmic consequence of Theorems 6 and 10 are not restricted in the design of sub-exponential parameterized algorithms (i.e., Theorems 7 and 11). Bidimensionality theory had meta-algorithmic applications in the automatic derivation of linear-time kernels for wide families of parameterized problems [12, 51]. Apart from its applications to parameterized complexity, Bidimensionality Theory was also used for the automated design of Fast Polynomial Time Approximation Schemes (FPTAS) in [28] and [48].

Proving extensions of Theorems 7 and 11 for wider families of graph classes (possibly with worse – but still moderate – time bounds) is an open challenge in parameterized algorithm design. For this, one may either need to find extensions of Theorems 6 and 10 for graph classes that are wider than H -minor free and apex-minor free graphs respectively (see [50] for an important step in this direction) or to invent alternative notions of grid-like structures whose presence in a graph is still able to certify a big enough value for the parameter \mathbf{p} (see [85, 101] and the end of Subsection 4.1).

5 The Irrelevant Vertex Technique

One of the most powerful tools in parameterized algorithm design is the *irrelevant vertex technique*, introduced in [108] in order to derive (among others) FPT-algorithms for the H -MINOR CHECKING (Theorem 2) and the k -DISJOINT PATHS Problem. The formal definition of the latter is the following.

k -DISJOINT PATHS

Instance: A graph G and a sequence of pairs
terminals $T = (s_1, t_1), \dots, (s_k, t_k) \in (V(G) \times V(G))^k$.

Parameter: k .

Question: Are there k pairwise vertex disjoint paths
 P_1, \dots, P_k in G such that for every $i \in \{1, \dots, k\}$,
 P_i has endpoints s_i and t_i ?

We stress that, in [108], both H -MINOR CHECKING and k -DISJOINT PATHS were treated simultaneously and the methodology that we present below is similar for both of them. In this section we give an outline of the $O_k(n^3)$ algorithm in [108] for the k -DISJOINT PATHS problem and we present some of the most important combinatorial results that supported the proof of its correctness.

5.1 The General Framework

Given an instance (G, T, k) of the k -DISJOINT PATHS problem, we say that a vertex $v \in V(G)$ is an *irrelevant* vertex of G if (G, T, k) and $(G \setminus v, T, k)$ are equivalent instances of the problem.

The general scheme of the algorithm in [108] is the following:

Irrelevant Vertex for the class \mathcal{G}_k

Input: An instance (G, T, k) of k -DISJOINT PATHS

Output: A (reduced) equivalent instance of k -DISJOINT PATHS

1. **while** $G \notin \mathcal{G}_k$,
2. *find* an irrelevant vertex v in G
3. set $G \leftarrow G \setminus v$
4. **output** (G, T, k)

Clearly, each variant of the above scheme depends on the parameterized class \mathcal{G}_k and creates an equivalent instance that belongs to \mathcal{G}_k . The algorithm in [108] applies the above scheme in two *phases*: the first phase considers

$$\mathcal{G}_k = \{G \mid G \text{ is a } K_{h(k)}\text{-minor free graph}\}$$

for some recursive function h and produces equivalent instances where the input graph does not contain a “big clique” as a minor. The second phase assumes that the input graph excludes such a clique and considers

$$\mathcal{G}_k = \{G \mid G \text{ is a } G_{g(k)}\text{-minor free graph}\},$$

for some recursive function $g : \mathbb{N} \rightarrow \mathbb{N}$. This produces an equivalent instance that, from Theorem 5, has treewidth bounded by $O_k(1)$ and, in this case, the problem can be solved in $O_k(n)$ steps, using dynamic programming or, alternatively, by just using Courcelle’s theorem.

It now remains to explain how Step 2 of the above scheme (i.e., finding an irrelevant vertex) is implemented in each of these two phases.

We omit the description of the first phase. Instead, we restrict ourselves to the second phase, as it encompasses the most combinatorially rich part of [108]. We just mention that the function h is determined from the results in [108] on the correctness of the first phase. The function g will be defined in the course of the description of the second phase below.

Assume now that we have an instance (G, T, k) of k -DISJOINT PATHS where G excludes a clique $K_{h(k)}$ as a minor but, however, it still contains a the grid $G_{g(k)}$ as a minor which means that $\mathbf{tw}(G) \geq g(k)$. A big part of [108] is devoted to the characterization of such graphs, i.e., of H -minor free graphs with “big” treewidth. In particular, a major achievement of [108] was to show the Weak Structure Theorem of GMT, stating that such graphs contain some portion that is, in a sense, “almost flat”. At this point we postpone the description of the irrelevant vertex technique to Subsection 5.3 in order to give the precise statement of this theorem.

5.2 The Weak Structure Graph Minors Theorem

Walls. A wall of height k , $k \geq 1$, is the graph obtained from a $((k+1) \times (2 \cdot k + 2))$ -grid with vertices (x, y) , $x \in \{1, \dots, 2 \cdot k + 2\}$, $y \in \{1, \dots, k + 1\}$, by the removal of the “vertical” edges $\{(x, y), (x, y + 1)\}$ for odd $x + y$, and then the removal of all vertices of degree 1. We denote such a wall by W_k . The *corners* of the wall W_k are the vertices $c_1 = (1, 1)$, $c_2 = (2 \cdot k + 1, 0)$, $c_3 = (2 \cdot k + 1 + (k + 1 \bmod 2), k + 1)$ and $c_4 = (1 + (k + 1 \bmod 2), k + 1)$. We let $C = \{c_1, c_2, c_3, c_4\}$. A *subdivided wall* W of height k is a graph obtained from W_k by replacing some of its edges by paths without common internal vertices. We call the resulting graph W a *subdivision* of W_k . The *perimeter* P of a subdivided wall is the cycle defined by its boundary. The *layers* of a subdivided wall W of height k are recursively defined as follows. The first layer of W is its perimeter. For $i = 2, \dots, \lfloor \frac{k}{2} \rfloor$, the i -th layer of W is the $(i - 1)$ -th layer of the subwall W' obtained from W by removing from W its perimeter and all occurring vertices of degree 1 (see Figure 4).

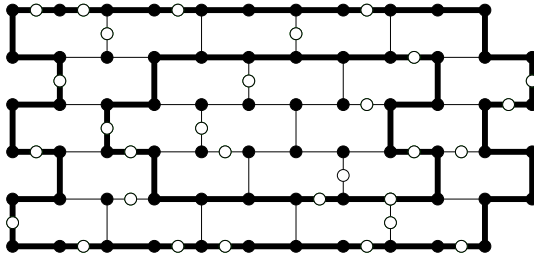


Fig. 4. A subdivided wall of height 5 and its two first layers. The first layer is its boundary

Compasses and Rural Divisions. Let W be a subdivided wall in G . Let K' be the connected component of $G \setminus P$ that contains $W \setminus P$. The *compass* K of W in G is the graph $G[V(K') \cup V(P)]$. Observe that W is a subgraph of K and K is connected. We say that a path of K is *perimetric* if its endpoints lie in the perimeter P of W . Let P_1 and P_2 be two perimetric paths of K with endpoints a_1, b_1 and a_2, b_2 respectively. We say that P_1 and P_2 *cross* in K if (a_1, a_2, b_1, b_2) is the cyclic ordering of their endpoints in P . We say that a wall is *flat* in G if K does not contain any pair of crossing and vertex-disjoint perimetric paths.

If J is a subgraph of K , we denote by $\partial_K J$ the set of all vertices $v \in V(J)$ such that either $v \in C$ or v is incident with an edge of K that is not in J . A *rural division* \mathcal{D} of the compass K is a collection (D_1, D_2, \dots, D_m) of subgraphs of K with the following properties:

1. $\{E(D_1), E(D_2), \dots, E(D_m)\}$ is a partition of non-empty subsets of $E(K)$,
2. for $i, j \in [m]$, if $i \neq j$ then $\partial_K D_i \neq \partial_K D_j$ and $V(D_i) \cap V(D_j) = \partial_K D_i \cap \partial_K D_j$,

3. for each $i \in [m]$ and all $x, y \in \partial_K D_i$ there exists a (x, y) -path in D_i with no internal vertex in $\partial_K D_i$,
4. for each $i \in [m]$, $|\partial_K D_i| \leq 3$, and
5. the hypergraph $H_K = (\bigcup_{i \in [m]} \partial_K D_i, \{\partial_K D_i \mid i \in [m]\})$ can be embedded in a closed disk Δ such that c_1, c_2, c_3 and c_4 appear in this order on the boundary of Δ and for each hyperedge e of H_K there exist $|e|$ mutually vertex-disjoint paths between e and C in K .

We call the elements of \mathcal{D} *flaps*. A flap $D \in \mathcal{D}$ is *internal* if $V(D) \cap V(P) = \emptyset$. We can now state one of the main results in [108], known as the *Weak Structure Graph Minors theorem*.

Theorem 12 [108]). *There exist recursive functions $g_1 : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ and $g_2 : \mathbb{N} \rightarrow \mathbb{N}$, such that for every two graphs H and G and every $q \in \mathbb{N}$, one of the following holds:*

1. H is a minor of G ,
2. $\text{tw}(G) \leq g_1(q, r)$, where $r = |V(H)|$
3. $\exists X \subseteq V(G)$ with $|X| \leq g_2(r)$ such that $G \setminus X$ contains as a subgraph a flat subdivided wall W where W has height q and the compass of W has a rural division \mathcal{D} such that each internal flap of \mathcal{D} has treewidth at most $g_1(r, q)$.

While the statement of Theorem 12 above is somehow complicated, the intuition behind it is simpler. It says that when a graph excludes some “small” graph H as a minor and has “big enough” treewidth, it is enough to remove a “few” vertices from it, i.e., the vertices in X , and take a graph $G \setminus X$ where it is possible to detect a subdivided wall W that is situated in a “flat” territory inside its perimeter P . The part of G that is inside P is the compass K of W which can be seen as the union of a collection of graphs (flaps) that are tree-like (have bounded treewidth) and are “planted” in that territory. Theorem 12 was used also in [2, 24] with the name “the Trinity Lemma”. However, a more depictive alternative nomenclature might be the “Sunny Forest Lemma”, in the sense that the compass K is a forest, whose trees are the flaps, and X is the sun throwing its rays at it!

In [59], an optimized version of the above result was proved where $g_1(r, q) = O_r(q)$ and $g_2(r)$ is equal to the apex number of H , i.e., the minimum number of vertices that, when removed from H , leave a planar graph. This improved version can easily yield both Theorems 6 and 10. In case H is an apex graph, i.e., it can become planar with the removal of a single vertex, the result in [59] implies that $X = \emptyset$ which gives an analogue of Theorem 12 for apex minor-free graphs. As, in this case, the “sun” X does not exist, we are tempted to call this “apex”-variant of Theorem 12 the “Dark Forest Lemma”.

5.3 Irrelevant Vertices and Linkages

We now go back to the task of detecting an irrelevant vertex in a graph G that excludes $K_{h(k)}$ and has treewidth bigger than $g(k)$. Recall that, at this point, h

has already been determined so that the previous phase of the algorithm runs correctly. In what follows, we set $g(k) = g_1(f_0(k) \cdot f_1(\lambda(k)), h(k))$ where g_1 is the function in Theorem 12, $f_0(k) = \lceil \sqrt{2k} \rceil + 1$, and f_1 and λ will be determined later.

According to [108], it is possible, in $O_k(n^2)$ steps, to detect in G a set X and a subdivided wall W of height $q = f_0(k) \cdot f_1(\lambda(k))$ of $G \setminus X$ where $|X| \leq g_2(h(k))$, as indicated in Theorem 12. For simplicity, we restrict our presentation to the case where X is an empty set, i.e., $|X| = 0$. Even if the ideas for the more general case are of the same flavor, they are quite more complicated and we prefer to omit them here.

Using a counting argument based on the definition of f_0 , it is easy to see that W contains a subdivided wall W' of height $q' = f_1(\lambda(k))$ whose compass K' avoids all terminals of the pairs in T .

The next step of the algorithm in [108] is based on the claim that if we take q' to be “big enough”, then any vertex v_{mid} of the *inner* layer L_{in} of W' is an irrelevant vertex and therefore it can be safely removed from G . While such a vertex is easy to detect, to prove that it is indeed irrelevant – for some suitable choice of q' – is not easy. We just mention that papers XXI [111] and XXII [107] of the Graph Minor series were devoted to it. Below, we present only some basic notions and ideas used in this proof. For this, we first need the definition of a k -linkage, introduced in [111].

A k -linkage in a graph G is a set of k pairwise disjoint paths of it. The *endpoints* of a linkage \mathcal{L} are the endpoints of the paths in \mathcal{L} . The *pattern* of \mathcal{L} is defined as

$$\pi(\mathcal{L}) = \{\{s, t\} \mid \mathcal{L} \text{ contains a path from } s \text{ to } t\}$$

Two k -linkages are *equivalent* if they have the same pattern.

W.l.o.g. we assume that all terminals involved in T are distinct. This implies that every solution to the k -DISJOINT PATHS problem is a k -linkage, whose pattern is determined by the pairs in T . To prove the irrelevance of the vertex v_{mid} , it is enough to show that any linkage \mathcal{L} whose paths meet L_{in} can be replaced with an equivalent one that avoids it. To obtain an idea of how paths in \mathcal{L} may reside inside K' , we need to make some observations.

Let \mathcal{R} be the linkage defined by the connected components of $(\bigcup_{L \in \mathcal{L}} L) \cap K'$, i.e., the subpaths of the paths in \mathcal{L} that are “cropped” by the compass K' (notice that all paths in \mathcal{R} are perimetric). By the flatness of W' , it is not possible that two paths in \mathcal{R} cross in K' . Moreover, by the definition of the rural division \mathcal{D}' of K' , each layer of W' , different than the inner one, is a separator of G . Therefore, if a path in \mathcal{R} meets layers L_i and L_j for $i \leq j$, then it should also meet layer L_μ for every $\mu \in \{i, \dots, j\}$. These observations argue that, intuitively, paths in \mathcal{R} cross K' as if K' were a graph embedded in a disk bounded by P – see Figure 5 for a visualization of this. One may now claim that the infrastructure of a “big enough” subdivided wall W' should provide enough space inside K'

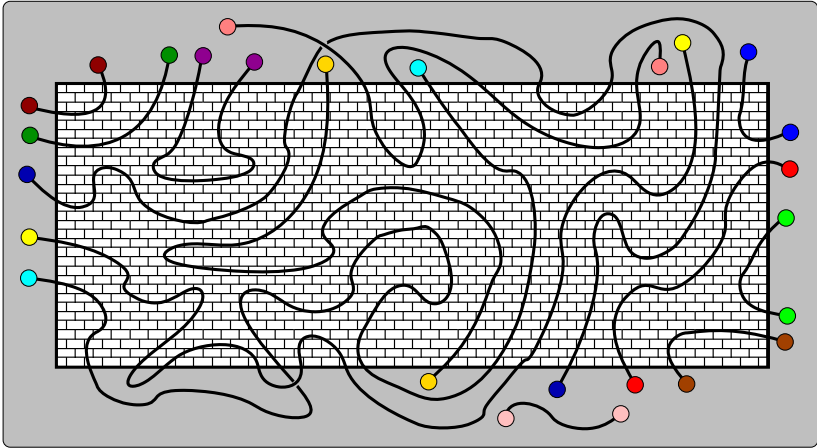


Fig. 5. A subdivided wall W' and the way a 13-linkage \mathcal{L} is traversing its compass K' . The only vertices that are depicted are the endpoints of the paths in \mathcal{L} (white vertices). The only edges that are depicted are those of the paths in \mathcal{L} and the edges of W' . The grey area contains the vertices and the edges of the graph G that do not belong to K' .

so that the paths of \mathcal{L} could be rerouted to an equivalent linkage that does not enter very deeply inside K' . To formalize this claim Robertson and Seymour defined the notion of a vital linkage in [111].

A linkage \mathcal{L} in a graph G is called *vital* if its vertices meet all the vertices of G and if there is no other linkage in G that is equivalent to \mathcal{L} . An example of a vital k -linkage in a graph is depicted in Figure 6. Clearly, if a solution of the k -DISJOINT PATHS Problem corresponds to a vital linkage, then no irrelevant vertex can be detected. The main result of [111] asserts that this possible “lack of flexibility” of linkages vanishes when graphs have big enough treewidth.

Theorem 13. *There exists a recursive function $\lambda : \mathbb{N} \rightarrow \mathbb{N}$ such that every graph with a vital k -linkage has treewidth at most $\lambda(k)$.*

Actually, it was also proved in [111] that treewidth can be replaced by pathwidth in Theorem 13. As the proof of 13 uses the Structure Theorem of the GMT [109], the upper bound for λ that follows from [111] is immense. However it was proved in [3] that in the case of planar graphs it holds that $\lambda(k) = 2^{O(k)}$. Moreover, this bound is, in a sense, tight: as argued in [3], for each k it is possible to construct a planar graph that contains a vital k -linkage and has treewidth $2^{\Omega(k)}$ (the 5-linkage in the graph of Figure 6 already gives the flavor of such a construction).

Let now G' be the subgraph of G defined by the union of the paths in \mathcal{L} , and the compass K' of W' . At this point, a naive idea might be to directly apply Theorem 13 and set $q' = \lambda(k)$ so that the linkage \mathcal{L} of G' , corresponding to a solution of the k -DISJOINT PATHS problem, cannot be vital. However, from this alone, we cannot expect nothing better than avoiding some vertices that will not

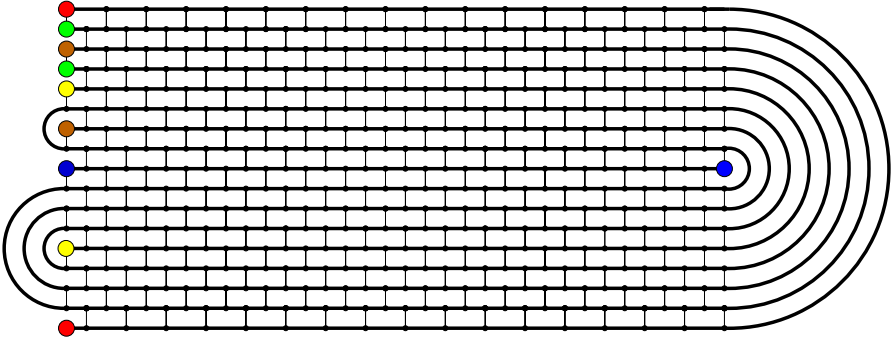


Fig. 6. A graph of treewidth 17 and a vital 5-linkage in it

necessarily be the vertices in L_{in} . Therefore, a non-vital linkage alone does not provide the flexibility we need in order to reroute in G' the paths of \mathcal{L} in a way that L_{in} is avoided.

Curiously, it appears that the importance of non-vital linkages is rather qualitative than quantitative. Based on their “elementary” flexibility, it is possible to prove that none of the paths in \mathcal{R} “bounces” much. In particular, \mathcal{L} can be chosen in a way that if a path in \mathcal{R} meets some layer L_i in two different vertices x and y , then its subpath between x and y will not meet any layer L_j where $|i - j| \geq f_1(\lambda(k))$, for some recursive function f_1 . This directly implies that paths in \mathcal{R} do not go deeper than layer $L_{f_1(\lambda(k))}$ and thus they avoid L_{in} when $q' = f_1(\lambda(k))$. That way, it is possible to prove what we need: if the height of W' is $f_1(\lambda(k))$, then another linkage, equivalent to \mathcal{L} exists in G' (and therefore in G as well) that avoids L_{in} .

We should stress that even if the above sketch might be “convincing” for a good-tempered reader, it is far from being a formal proof. In the more realistic case where X is non-empty, a more complicated criterion for the choice of the subdivided wall W' should be devised and a bigger lower bound for the height of W' is necessary so that it contains an irrelevant vertex. In fact, this requires bigger lower bounds for both f_0 and f_1 . The whole proof is quite technical and has been the main purpose of [107].

According to the above discussion, the second phase of the algorithm runs in $O_k(n^3)$ steps and outputs a graph of treewidth at most $g(k)$. As proved in [116], the k -DISJOINT PATHS problem can be solved by a $f_2(k) \cdot n$ step dynamic programming algorithm where $f_2(k) = 2^{O(k \log k)}$ (see [1, 90] for results related to this problem). As the parameter dependence of the running time of this last step is dominant in the running time of the algorithm, we conclude that the overall parameter dependence is:

$$O(f_2(g_1(f_0(k) \cdot f_1(\lambda(k))), h(k))).$$

Clearly, an improvement on the existing bounds for any of the functions g_1, h, f_0, f_1, f_2 , and λ would be an important step towards reducing the

parameter dependence of the algorithm for the k -DISJOINT PATHS problem. In fact, the only function that is “really immense” is λ , because the proof of its existence was based on the Structure Theorem of the GMT [109]. In this direction an alternative, relatively simpler, proof was given in [80] that avoids the core results of [109]. Using a rough estimation, the proof in [80] should give that that $\lambda(k) = 2^{2^{2^{O(k)}}}$ which changes the status of the parameter dependence in Robertson and Seymour’s algorithm from “immense” to “huge”. Clearly, any further improvements, even for special cases or variants of the problem, are highly welcome (see [3]).

5.4 Applications

The above description already outlines a powerful algorithmic framework that could not be of use for just one problem. Below, we mention a series of results in parameterized algorithms where the irrelevant vertex technique (or extensions of it) has been applied. We sort them in chronological order of their appearance.

- [24] A proof of the following meta-algorithmic result: *Let \mathcal{C} be a class of graphs excluding and h -vertex graph H as a minor. Then any first-order definable decision problem can be solved in time $O_{h+|\phi|}(n^{O(1)})$, where f is a computable function and ϕ is the sentence defining the decision problem.*
- [77] A $2^{O(g)} \cdot n$ step algorithm that, given a graph G and a non-negative integer g either outputs an embedding of G in a surface of genus g or a minor of G that belongs to $\mathbf{obs}_{\leq m}(\mathcal{G}_g)$ where \mathcal{G}_g contains all graphs embeddible in a surface of Euler genus g . A previous result of this type, but not with single-exponential parameter dependence appeared previously in [94].
- [2] A proof that it is possible to construct an algorithm that, given the obstruction sets of two minor-closed graph classes \mathcal{G}_1 and \mathcal{G}_2 , outputs the obstruction set of the class $\mathcal{G} = \mathcal{G}_1 \cup \mathcal{G}_2$. Also, in the same paper, it was proved that it is possible to construct an algorithm that given the obstruction set of a minor-closed graph class \mathcal{G} and a non-negative integer k , outputs the obstruction set of the class $k\text{-almost}(\mathcal{G}) = \{G \mid \exists S \subseteq V(G) : |S| \leq k \text{ and } G \setminus S \in \mathcal{G}\}$ (see also [19, 71, 93] for related results).
- [83] An $O_k(n^{O(1)})$ time algorithm for solving the INDUCED CYCLE THROUGH TERMINALS problem: *Given a graph G , embedded in some surface, and a set $S \subseteq V(G)$ of terminals, does G contain an induced cycle that meets all vertices in S ?*
- [60] An $2^{O(k^{3/2})} \cdot n^{O(1)}$ time algorithm for the ODD INDUCED CYCLE PACKING on planar graphs: *Given a graph G and an integer k , does G contains k induced odd cycles?*
- [79] An $O_k(n^{O(1)})$ time algorithm for the ODD CYCLE PACKING problem: *Given a graph G and an integer k , does G contains k vertex-disjoint odd cycles?*

- [64] An $O_k(n)$ time algorithm for the BIPARTITE CONTRACTION problem: *Given a graph G and an integer k , can we obtain a bipartite graph from G by a sequence of at most k edge contractions in G ?*
- [49] Subexponential $2^{O(\sqrt{k})} \cdot n$ time algorithms for the PARTIAL VERTEX COVER and PARTIAL DOMINATING SET problems for apex minor-free graphs: *Given a graph G and integers k, t , can we cover (resp. dominate) at least t edges (resp. vertices) with at most k vertices?*
- [61] An $O_k(n^3)$ algorithm for the TOPOLOGICAL MINOR CONTAINMENT and the IMMERSION CONTAINMENT problems: *Given two graphs G and H , where $n(H) = k$, does G contain H as a topological minor (resp. immersion). The results in [61] can be seen as a major extension of the algorithm in [108].*
- [45] A proof of the following result on kernelization: *Let \mathcal{G}_r be the class of all K_r -minor free graphs. Then the DOMINATING SET Problem and the CONNECTED DOMINATING SET problem, asking whether a graph G has a (connected) dominating set of size k , has a linear $O_r(k)$ -size kernel when restricted in graphs in \mathcal{G}_r .*
- [69] An $O_{k+g}(n^3)$ algorithm for the CONTRACTION CONTAINMENT problem restricted to graphs of Euler genus g . The CONTRACTION CONTAINMENT problem asks, with input two graphs G and H , where $n(H) = k$, whether H is a contraction of G .

Clearly, the above list is just indicative and is expected to grow more. Further algorithmic applications of the weak structure theorem and/or the irrelevant vertex technique can be found in [66, 68, 70, 72, 73, 75, 76, 78]. Also results where the irrelevant vertex idea is applied in a more general sense, without using directly results of the GMT, can be found in [25, 47, 63, 92].

6 Conclusions

Covering the whole range of the contributions of the GMT to the design of parameterized algorithms is a task that cannot fit in the space of this short presentation. The progress over the last years towards building an Algorithmic Graph Minors Theory has been noticeable and we believe that there is much more “algorithmic material” to be extracted from this deep and fascinating theory. As we expect more results to emerge from GMT, not only in parameterized algorithms but also in other fields of algorithm design, we hope that this small portion of the material covered will be of use as an invitation to this direction.

Acknowledgements. We wish to thank Isolde Adler, Marcin Kamiński and Stavros G. Kolliopoulos for their detailed comments, remarks, and suggestions on this text. We also wish to personally thank Michael R. Fellows for his support and encouragement while investigating this “minor” corner of parameterized algorithms.

References

1. Adler, I., Dorn, F., Fomin, F.V., Sau, I., Thilikos, D.M.: Fast Minor Testing in Planar Graphs. In: de Berg, M., Meyer, U. (eds.) ESA 2010. LNCS, vol. 6346, pp. 97–109. Springer, Heidelberg (2010)
2. Adler, I., Grohe, M., Kreutzer, S.: Computing excluded minors. In: Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2008, pp. 641–650. Society for Industrial and Applied Mathematics, Philadelphia (2008)
3. Adler, I., Kolliopoulos, S.G., Krause, P.K., Lokshtanov, D., Saurabh, S., Thilikos, D.: Tight Bounds for Linkages in Planar Graphs. In: Aceto, L., Henzinger, M., Sgall, J. (eds.) ICALP 2011, Part I. LNCS, vol. 6755, pp. 110–121. Springer, Heidelberg (2011)
4. Alber, J., Bodlaender, H.L., Fernau, H., Kloks, T., Niedermeier, R.: Fixed parameter algorithms for dominating set and related problems on planar graphs. *Algorithmica* 33(4), 461–493 (2002)
5. Alber, J., Dorn, F., Niedermeier, R.: Experimental evaluation of a tree decomposition-based algorithm for vertex cover on planar graphs. *Discrete Applied Mathematics* 145(2), 219–231 (2005); *Structural Decompositions, Width Parameters, and Graph Labelings*
6. Alber, J., Niedermeier, R.: Improved Tree Decomposition Based Algorithms for Domination-like Problems. In: Rajsbaum, S. (ed.) LATIN 2002. LNCS, vol. 2286, pp. 613–627. Springer, Heidelberg (2002)
7. Amir, E.: Approximation algorithms for treewidth. *Algorithmica* 56, 448–479 (2010)
8. Arnborg, S., Lagergren, J., Seese, D.: Easy problems for tree-decomposable graphs. *Journal of Algorithms* 12, 308–340 (1991)
9. Betzler, N., Niedermeier, R., Uhlmann, J.: Tree decompositions of graphs: Saving memory in dynamic programming. *Discrete Optimization* 3(3), 220–229 (2006); *Graphs and Combinatorial Optimization*
10. Bienstock, D., Dean, N.: On obstructions to small face covers in planar graphs. *J. Combin. Theory Ser. B* 55(2), 163–189 (1992)
11. Björklund, A., Husfeldt, T., Kaski, P., Koivisto, M.: Fourier meets Möbius: fast subset convolution. In: STOC, pp. 67–74. ACM (2007)
12. Bodlaender, H., Fomin, F., Lokshtanov, D., Penninkx, E., Saurabh, S., Thilikos, D.: (Meta) kernelization. In: 50th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2009). ACM (2009)
13. Bodlaender, H.L.: Dynamic Programming on Graphs with Bounded Treewidth. In: Lepistö, T., Salomaa, A. (eds.) ICALP 1988. LNCS, vol. 317, pp. 105–118. Springer, Heidelberg (1988)
14. Bodlaender, H.L.: A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput.* 25(6), 1305–1317 (1996)
15. Bodlaender, H.L., Fellows, M.R., Hallett, M.T.: Beyond NP-completeness for problems of bounded width: hardness for the W hierarchy. In: Twenty-sixth Annual ACM Symposium on Theory of Computing (STOC 1994), pp. 449–458. ACM, New York (1994)
16. Bodlaender, H.L., Telle, J.A.: Space-efficient construction variants of dynamic programming. *Nordic J. Comput.* 11(4), 374–385 (2004)
17. Borie, R.B., Parker, R.G., Tovey, C.A.: Automatic generation of linear-time algorithms from predicate calculus descriptions of problems on recursively constructed graph families. *Algorithmica* 7, 555–581 (1992)

18. Cai, L., Juedes, D.: On the existence of subexponential parameterized algorithms. *J. Comput. System Sci.* 67(4), 789–807 (2003)
19. Cattell, K., Dinneen, M.J., Downey, R.G., Fellows, M.R., Langston, M.A.: On computing graph minor obstruction sets. *Theor. Comput. Sci.* 233, 107–127 (2000)
20. Chlebíková, J.: The structure of obstructions to treewidth and pathwidth. *Discrete Applied Mathematics* 120(1-3), 61–71 (2002)
21. Courcelle, B., Downey, R.G., Fellows, M.R.: A note on the computability of graph minor obstruction sets for monadic second order ideals. In: *First Japan–New Zealand Workshop on Logic in Computer Science*, Auckland, vol. 3, pp. 1194–1198 (1997) (electronic)
22. Cygan, M., Nederlof, J., Pilipczuk, M., Pilipczuk, M., van Rooij, J.M.M., Woźtaszczyk, J.O.: Solving connectivity problems parameterized by treewidth in single exponential time. In: *IEEE 52nd Annual Symposium on Foundations of Computer Science (FOCS 2011)*, pp. 150–159. IEEE Computer Society (2011)
23. Dahan, X., Tillich, J.-P.: Ramanujan graphs of very large girth based on octonions. CoRR, arXiv:1011.2642 (November 2010–2011)
24. Dawar, A., Grohe, M., Kreutzer, S.: Locally excluding a minor. In: *21st IEEE Symposium on Logic in Computer Science (LICS 2007)*, pp. 270–279. IEEE, New York (2007)
25. Dawar, A., Kreutzer, S.: Domination problems in nowhere-dense classes. In: *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FST-TCS 2009)*, pp. 157–168 (2009)
26. Demaine, E.D., Fomin, F.V., Hajiaghayi, M., Thilikos, D.M.: Bidimensional parameters and local treewidth. *SIAM J. Discrete Math.* 18(3), 501–511 (2004) (electronic)
27. Demaine, E.D., Fomin, F.V., Hajiaghayi, M., Thilikos, D.M.: Subexponential parameterized algorithms on graphs of bounded genus and H -minor-free graphs. *Journal of the ACM* 52(6), 866–893 (2005)
28. Demaine, E.D., Hajiaghayi, M.: Bidimensionality: new connections between FPT algorithms and PTASs. In: *Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 590–601. ACM, New York (2005) (electronic)
29. Demaine, E.D., Hajiaghayi, M.: Linearity of grid minors in treewidth with applications through bidimensionality. *Combinatorica* 28(1), 19–36 (2008)
30. Demaine, E.D., Hajiaghayi, M., Kawarabayashi, K.: Algorithmic graph minor theory: Improved grid minor bounds and Wagner’s contraction. *Algorithmica* 54(2), 142–180 (2009)
31. Demaine, E.D., Hajiaghayi, M., Thilikos, D.M.: The bidimensional theory of bounded-genus graphs. *SIAM J. Discrete Math.* 20(2), 357–371 (2006)
32. Diestel, R.: *Graph Theory*, 3rd edn. Graduate Texts in Mathematics, vol. 173. Springer (2005)
33. Diestel, R., Jensen, T.R., Gorbunov, K.Y., Thomassen, C.: Highly connected sets and the excluded grid theorem. *J. Combin. Theory Ser. B* 75(1), 61–73 (1999)
34. Dinneen, M.J.: Too many minor order obstructions (for parameterized lower ideals). In: *First Japan–New Zealand Workshop on Logic in Computer Science*, Auckland, vol. 3(11), pp. 1199–1206 (1997) (electronic)
35. Dorn, F.: Dynamic Programming and Fast Matrix Multiplication. In: Azar, Y., Erlebach, T. (eds.) *ESA 2006*. LNCS, vol. 4168, pp. 280–291. Springer, Heidelberg (2006)
36. Dorn, F., Fomin, F.V., Thilikos, D.M.: Fast Subexponential Algorithm for Non-local Problems on Graphs of Bounded Genus. In: Arge, L., Freivalds, R. (eds.) *SWAT 2006*. LNCS, vol. 4059, pp. 172–183. Springer, Heidelberg (2006)

37. Dorn, F., Fomin, F.V., Thilikos, D.M.: Catalan structures and dynamic programming in H -minor-free graphs. In: ACM-SIAM Symposium on Discrete Algorithms (SODA 2008), pp. 631–640. SIAM (2008)
38. Dorn, F., Penninkx, E., Bodlaender, H.L., Fomin, F.V.: Efficient exact algorithms on planar graphs: Exploiting sphere cut decompositions. *Algorithmica* 58(3), 790–810 (2010)
39. Downey, R.G., Fellows, M.R.: Parameterized complexity. Monographs in Computer Science. Springer, New York (1999)
40. Ellis, J.A., Sudborough, I.H., Turner, J.S.: The vertex separation and search number of a graph. *Information and Computation* 113(1), 50–79 (1994)
41. Erdős, P., Sachs, H.: Reguläre graphen gegebener tailenweite mit minimaler knollenzahh. *Wiss. Z. Univ. Halle-Willenberg Math. Nat.* 12, 251–258 (1063)
42. Fellows, M.: Towards Fully Multivariate Algorithmics: Some New Results and Directions in Parameter Ecology. In: Fiala, J., Kratochvíl, J., Miller, M. (eds.) IWOCA 2009. LNCS, vol. 5874, pp. 2–10. Springer, Heidelberg (2009)
43. Fellows, M.R., Langston, M.A.: On search, decision, and the efficiency of polynomial-time algorithms. *J. Comput. System Sci.* 49(3), 769–779 (1994)
44. Flum, J., Grohe, M.: Parameterized Complexity theory. Texts in Theoretical Computer Science. An EATCS Series. Springer, Berlin (2006)
45. Fomin, F.V., Daniel Lokshtanov, S.S., Thilikos, D.M.: Linear kernels for (connected) dominating set on h -minor-free graphs. In: 23st ACM-SIAM Symposium on Discrete Algorithms (SODA 2012). ACM-SIAM, San Francisco (2012)
46. Fomin, F.V., Golovach, P.A., Thilikos, D.M.: Contraction obstructions for treewidth. *J. Comb. Theory, Ser. B* 101(5), 302–314 (2011)
47. Fomin, F.V., Lokshtanov, D., Misra, N., Philip, G., Saurabh, S.: Hitting forbidden minors: Approximation and kernelization. In: STACS, pp. 189–200 (2011)
48. Fomin, F.V., Lokshtanov, D., Raman, V., Saurabh, S.: Bidimensionality and EP-TAS. In: 22st ACM-SIAM Symposium on Discrete Algorithms (SODA 2011), pp. 748–759. ACM-SIAM, San Francisco (2011)
49. Fomin, F.V., Lokshtanov, D., Raman, V., Saurabh, S.: Subexponential algorithms for partial cover problems. *Inf. Process. Lett.* 111(16), 814–818 (2011)
50. Fomin, F.V., Lokshtanov, D., Saurabh, S.: Bidimensionality and geometric graphs. In: 23st ACM-SIAM Symposium on Discrete Algorithms (SODA 2012). ACM-SIAM, San Francisco (2012)
51. Fomin, F.V., Lokshtanov, D., Saurabh, S., Thilikos, D.M.: Bidimensionality and kernels. In: 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2010), Austin, Texas, pp. 503–510. ACM-SIAM (2010)
52. Fomin, F.V., Thilikos, D.M.: Dominating sets in planar graphs: branch-width and exponential speed-up. *SIAM J. Comput.* 36(2), 281–309 (2006) (electronic)
53. Frick, M., Grohe, M.: The complexity of first-order and monadic second-order logic revisited. *Ann. Pure Appl. Logic* 130(1-3), 3–31 (2004)
54. Friedman, H., Robertson, N., Seymour, P.D.: The metamathematics of the graph minor theorem. In: *Logic and Combinatorics* (Arcata, Calif., 1985). *Contemp. Math.*, vol. 65, pp. 229–261. Amer. Math. Soc., Providence (1987)
55. Friedman, H.M.: Internal finite tree embeddings. In: *Reflections on the foundations of mathematics* (Stanford, CA, 1998). *Lect. Notes Log.*, vol. 15, pp. 60–91. Assoc. Symbol. Logic, Urbana (2002)
56. Gavril, F.: The intersection graphs of subtrees in trees are exactly the chordal graphs. *J. Combin. Theory, Ser. B* 16(1), 47–56 (1974)

57. Geelen, J.F., Gerards, A.M.H., Robertson, N., Whittle, G.P.: On the excluded minors for the matroids of branch-width k . *J. Combin. Theory Ser. B* 88(2), 261–265 (2003)
58. Giannopoulou, A.C., Thilikos, D.M.: Obstructions for tree-depth. *Electronic Notes in Discrete Mathematics* 34, 249–253 (2009)
59. Giannopoulou, A.C., Thilikos, D.M.: Optimizing the graph minors weak structure theorem. *CoRR*, arXiv:1102.5762 (February 2011)
60. Golovach, P.A., Kamiński, M., Paulusma, D., Thilikos, D.M.: Induced Packing of Odd Cycles in a Planar Graph. In: Dong, Y., Du, D.-Z., Ibarra, O. (eds.) *ISAAC 2009*. LNCS, vol. 5878, pp. 514–523. Springer, Heidelberg (2009)
61. Grohe, M., Kawarabayashi, K., Marx, D., Wollan, P.: Finding topological subgraphs is fixed-parameter tractable. In: *Proceedings of the 43rd ACM Symposium on Theory of Computing (STOC 2011)*, pp. 479–488 (2011)
62. Gu, Q.-P., Tamaki, H.: Improved Bounds on the Planar Branchwidth with Respect to the Largest Grid Minor Size. In: Cheong, O., Chwa, K.-Y., Park, K. (eds.) *ISAAC 2010, Part II*. LNCS, vol. 6507, pp. 85–96. Springer, Heidelberg (2010)
63. Heggernes, P., van’t Hof, P., Jansen, B.M.P., Kratsch, S., Villanger, Y.: Parameterized Complexity of Vertex Deletion into Perfect Graph Classes. In: Owe, O., Steffen, M., Telle, J.A. (eds.) *FCT 2011*. LNCS, vol. 6914, pp. 240–251. Springer, Heidelberg (2011)
64. Heggernes, P., van’t Hof, P., Lokshantov, D., Paul, C.: Obtaining a bipartite graph by contracting few edges. In: *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2011)*, pp. 217–228 (2011)
65. Higman, G.: Ordering by divisibility in abstract algebras. *Proc. London Math. Soc.* 2(3), 326–336 (1952)
66. Ito, T., Kamiński, M., Paulusma, D., Thilikos, D.M.: Parameterizing cut sets in a graph by the number of their components. *Theor. Comput. Sci.* 412(45), 6340–6350 (2011)
67. Johnson, D.S.: The NP-completeness column: An ongoing guide. *Journal of Algorithms* 8(2), 285–303 (1987)
68. Kamiński, M., Nishimura, N.: Finding an induced path of given parity in planar graphs in polynomial time. In: *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2011)*, pp. 656–670. ACM (2012)
69. Kamiński, M., Thilikos, D.M.: Contraction checking in graphs on surfaces. In: *29th International Symposium on Theoretical Aspects of Computer Science (STACS 2012)*, pp. 182–193 (2012)
70. Kawarabayashi, K.: Half integral packing, Erdős-Pósa-property and graph minors. In: *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2007*, pp. 1187–1196. Society for Industrial and Applied Mathematics, Philadelphia (2007)
71. Kawarabayashi, K.: Planarity allowing few error vertices in linear time. In: *50th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2009*, pp. 639–648 (2009)
72. Kawarabayashi, K., Kobayashi, Y.: The edge disjoint paths problem in eulerian graphs and 4-edge-connected graphs. In: *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010*, pp. 345–353. Society for Industrial and Applied Mathematics, Philadelphia (2010)
73. Kawarabayashi, K., Kobayashi, Y.: An improved algorithm for the half-disjoint paths problem. *SIAM J. Discrete Math.* 25(3), 1322–1330 (2011)

74. Kawarabayashi, K., Kobayashi, Y., Reed, B.: The disjoint paths problem in quadratic time. *Journal of Combinatorial Theory, Series B* (2011)
75. Kawarabayashi, K., Kreutzer, S., Mohar, B.: Linkless and flat embeddings in 3-space and the unknot problem. In: *Proceedings of the 2010 Annual Symposium on Computational Geometry, SoCG 2010*, pp. 97–106. ACM, New York (2010)
76. Kawarabayashi, K., Li, Z., Reed, B.A.: Recognizing a totally odd K_4 -subdivision, parity 2-disjoint rooted paths and a parity cycle through specified elements. In: *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010*, pp. 318–328 (2010)
77. Kawarabayashi, K., Mohar, B., Reed, B.A.: A simpler linear time algorithm for embedding graphs into an arbitrary surface and the genus of graphs of bounded tree-width. In: *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008*, pp. 771–780 (2008)
78. Kawarabayashi, K., Reed, B.A.: Hadwiger’s conjecture is decidable. In: *41st Annual ACM Symposium on Theory of Computing (STOC 2009)*, pp. 445–454 (2009)
79. Kawarabayashi, K., Reed, B.A.: Odd cycle packing. In: *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010*, pp. 695–704 (2010)
80. Kawarabayashi, K., Wollan, P.: A shorter proof of the graph minor algorithm: the unique linkage theorem. In: *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008*, pp. 771–780 (2008)
81. Kneis, J., Langer, A.: A practical approach to Courcelle’s theorem. *Electron. Notes Theor. Comput. Sci.* 251, 65–81 (2009)
82. Kneis, J., Langer, A., Rossmanith, P.: Courcelle’s theorem - a game-theoretic approach. *CoRR*, arXiv:1104.3905 (April 2011)
83. Kobayashi, Y., Kawarabayashi, K.: Algorithms for finding an induced cycle in planar graphs and bounded genus graphs. In: *20th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2009)*, pp. 1146–1155. ACM-SIAM (2009)
84. Koutsonas, A., Thilikos, D.M., Yamazaki, K.: Outerplanar obstructions for matroid pathwidth. In: *EuroComb 2011: European Conference on Combinatorics, Graph Theory and Applications* (2011)
85. Kreutzer, S., Tazari, S.: On brambles, grid-like minors, and parameterized intractability of monadic second-order logic. In: *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010*, pp. 354–364. Society for Industrial and Applied Mathematics, Philadelphia (2010)
86. Kruskal, J.B.: Well-quasi-ordering, the tree theorem, and Vazsonyi’s conjecture. *Trans. Amer. Math. Soc.* 95, 210–225 (1960)
87. Kuratowski, K.: Sur le problème des courbes gauches en topologie. *Fund. Math.* 15, 271–283 (1930)
88. Lagergren, J.: Efficient parallel algorithms for graphs of bounded tree-width. *Journal of Algorithms* 20(1), 20–44 (1996)
89. Lagergren, J.: Upper bounds on the size of obstructions and intertwines. *J. Combin. Theory, Ser. B* 73, 7–40 (1998)
90. Lokshtanov, D., Marx, D., Saurabh, S.: Slightly superexponential parameterized problems. In: *22st ACM-SIAM Symposium on Discrete Algorithms (SODA 2011)*, pp. 760–776 (2011)
91. Lovász, L.: Graph minor theory. *Bull. Amer. Math. Soc. (N.S.)* 43(1), 75–86 (2006) (electronic)
92. Marx, D.: Chordal deletion is fixed-parameter tractable. *Algorithmica* 57(4), 747–768 (2010)
93. Marx, D., Schlotter, I.: Obtaining a planar graph by vertex deletion. *Algorithmica* 62(3-4), 807–822 (2012)

94. Mohar, B.: A linear time algorithm for embedding graphs in an arbitrary surface. *SIAM J. Discrete Math.* 12(1), 6–26 (1999)
95. Nash-Williams, C.S.J.A.: On well-quasi-ordering finite trees. *Proc. Cambridge Philos. Soc.* 59, 833–835 (1963)
96. Neumann, B.H.: On ordered division rings. *Trans. Amer. Math. Soc.* 66, 202–252 (1949)
97. Niedermeier, R.: Invitation to fixed-parameter algorithms. *Oxford Lecture Series in Mathematics and its Applications*, vol. 31. Oxford University Press, Oxford (2006)
98. Parsons, T.D.: Pursuit-evasion in a graph. In: *Proceedings Internat. Conf., Western Mich. Univ., Kalamazoo, Mich., 1976, Theory and Applications of Graphs. Lecture Notes in Math.*, vol. 642, pp. 426–441. Springer, Berlin (1978)
99. Ramachandramurthi, S.: The structure and number of obstructions to treewidth. *SIAM J. Discrete Math.* 10(1), 146–157 (1997)
100. Reed, B.A.: Finding approximate separators and computing tree width quickly. In: *Twenty-Fourth Annual ACM Symposium on Theory of Computing (STOC 1992)*, pp. 221–228. ACM Press (1992)
101. Reed, B.A., Wood, D.R.: Polynomial treewidth forces a large grid-like-minor. *Eur. J. Comb.* 33(3), 374–379 (2012)
102. Robertson, N., Seymour, P.D.: Graph minors. I. excluding a forest. *J. Combin. Theory, Ser. B* 35, 39–61 (1983)
103. Robertson, N., Seymour, P.D.: Graph minors. III. Planar tree-width. *J. Combin. Theory, Ser. B* 36(1), 49–64 (1984)
104. Robertson, N., Seymour, P.D.: Graph minors. II. algorithmic aspects of tree-width. *Journal of Algorithms* 7, 309–322 (1986)
105. Robertson, N., Seymour, P.D.: Graph minors. V. Excluding a planar graph. *J. Combin. Theory Ser. B* 41(1), 92–114 (1986)
106. Robertson, N., Seymour, P.D.: Graph minors. X. Obstructions to tree-decomposition. *J. Combin. Theory Ser. B* 52(2), 153–190 (1991)
107. Robertson, N., Seymour, P.D.: Graph minors. XXII. Irrelevant vertices in linkage problems (1992) (preprint)
108. Robertson, N., Seymour, P.D.: Graph minors. XIII. The disjoint paths problem. *J. Combin. Theory, Ser. B* 63(1), 65–110 (1995)
109. Robertson, N., Seymour, P.D.: Graph minors. XVI. Excluding a non-planar graph. *J. Combin. Theory Series B* 77, 1–27 (1999)
110. Robertson, N., Seymour, P.D.: Graph minors. XX. Wagner’s conjecture. *J. Combin. Theory Ser. B* 92(2), 325–357 (2004)
111. Robertson, N., Seymour, P.D.: Graph minors. XXI. Graphs with unique linkages. *J. Combin. Theory Ser. B* 99(3), 583–616 (2009)
112. Robertson, N., Seymour, P.D.: Graph minors XXIII. Nash-Williams’ immersion conjecture. *J. Combin. Theory Ser. B* 100(2), 181–205 (2010)
113. Robertson, N., Seymour, P.D., Thomas, R.: Quickly excluding a planar graph. *J. Combin. Theory Ser. B* 62(2), 323–348 (1994)
114. Rué, J., Sau, I., Thilikos, D.M.: Dynamic Programming for Graphs on Surfaces. In: *Abramsky, S., Gavoille, C., Kirchner, C., Meyer auf der Heide, F., Spirakis, P.G. (eds.) ICALP 2010, Part I. LNCS*, vol. 6198, pp. 372–383. Springer, Heidelberg (2010)
115. Rué, J., Stavropoulos, K.S., Thilikos, D.M.: Outerplanar obstructions for the feedback vertex set. *Electronic Notes in Discrete Mathematics* 34, 167–171 (2009)

116. Scheffler, P.: A practical linear time algorithm for disjoint paths in graphs with bounded tree-width. Technical Report 396/1994, FU Berlin, Fachbereich 3 Mathematik (1994)
117. Takahashi, A., Ueno, S., Kajitani, Y.: Minimal acyclic forbidden minors for the family of graphs with bounded path-width. *Disc. Math.* 127(1-3), 293–304 (1994); *Graph theory and applications, Hakone* (1990)
118. Thilikos, D.M.: Algorithms and obstructions for linear-width and related search parameters. *Discrete Applied Mathematics* 105, 239–271 (2000)
119. van Leeuwen, J.: Graph algorithms. In: *Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity (A)*, pp. 525–631. Elsevier Science (1990)
120. van Rooij, J.M.M., Bodlaender, H.L., Rossmanith, P.: Dynamic Programming on Tree Decompositions Using Generalised Fast Subset Convolution. In: Fiat, A., Sanders, P. (eds.) *ESA 2009. LNCS*, vol. 5757, pp. 566–577. Springer, Heidelberg (2009)
121. Wagner, K.: Über eine eigenschaft der ebenen komplexe. *Mathematische Annalen* 114, 570–590 (1937), 10.1007/BF01594196
122. Weiss, A.: Girths of bipartite sextet graphs. *Combinatorica* 4(2-3), 241–245 (1984)

Constraint Satisfaction Problems Parameterized above or below Tight Bounds: A Survey

Gregory Gutin¹ and Anders Yeo²

¹ Royal Holloway, University of London
Egham, Surrey, TW20 0EX, UK
gutin@cs.rhul.ac.uk

² University of Johannesburg
Auckland Park, 2006 South Africa
anders.yeo.work@gmail.com

This paper is dedicated to the 60th Birthday of Michael R. Fellows

Abstract. We consider constraint satisfaction problems parameterized above or below tight bounds. One example is MaxSat parameterized above $m/2$: given a CNF formula F with m clauses, decide whether there is a truth assignment that satisfies at least $m/2+k$ clauses, where k is the parameter. Among other problems we deal with are MaxLin2-AA (given a system of linear equations over \mathbb{F}_2 in which each equation has a positive integral weight, decide whether there is an assignment to the variables that satisfies equations of total weight at least $W/2+k$, where W is the total weight of all equations), Max- r -Lin2-AA (the same as MaxLin2-AA, but each equation has at most r variables, where r is a constant) and Max- r -Sat-AA (given a CNF formula F with m clauses in which each clause has at most r literals, decide whether there is a truth assignment satisfying at least $\sum_{i=1}^m (1-2^{r_i})+k$ clauses, where k is the parameter, r_i is the number of literals in Clause i , and r is a constant). We also consider Max- r -CSP-AA, a natural generalization of both Max- r -Lin2-AA and Max- r -Sat-AA, order (or, permutation) constraint satisfaction problems of arities 2 and 3 parameterized above the average value and some other problems related to MaxSat. We discuss results, both polynomial kernels and parameterized algorithms, obtained for the problems mainly in the last few years as well as some open questions.

1 Introduction

This paper surveys mainly recent results in a subarea of parameterized algorithms and complexity that was launched quite early in the short history of parameterized algorithms and complexity, namely, in the Year 2 BDF¹.

¹ BDF stands for Before Downey-Fellows, i.e., before 1999 when the first monograph describing foundations of parameterized algorithms and complexity was published [18].

Consider the well-known problem MAXSAT, where for a given CNF formula F with m clauses, we are asked to determine the maximum number of clauses of F that can be satisfied by a truth assignment. It is well-known (and shown below, in Section 4) that there exists a truth assignment to the variables of F which satisfies at least $m/2$ clauses.

The standard parametrization k -MAXSAT of MAXSAT is as follows: decide whether there is a truth assignment which satisfies at least k clauses of F , where k is the parameter. (We provide basic terminology and notation on parameterized algorithms and complexity in Section 2.) It is very easy to see that k -MAXSAT has a kernel with a linear number of clauses. Indeed, consider an instance I of k -MAXSAT. If $k \leq m/2$ then I is a YES-instance. Otherwise, we have $k > m/2$ and $m \leq 2k - 1$. Suppose that we managed somehow to obtain a better result, a kernel with at most pk clauses, where $1 \leq p < 2$. Is such a kernel of any interest? Such a kernel would be of interest only for $k > m/2$, i.e., when the size of the kernel would be bounded by $pk > pm/2$. Thus, such a kernel should be viewed as *huge* rather than *small* as the bound pk might suggest at the first glance.

The bound $m/2$ is tight as we can satisfy only half clauses in the instances consisting of pairs $(x), (\bar{x})$ of clauses. This suggests the following parameterization of MAXSAT *above tight bound* introduced by Mahajan and Raman [44]: decide whether there is a truth assignment which satisfies at least $m/2 + k$ clauses of F , where k is the parameter.

To the best of our knowledge, [44] was the first paper on problems parameterized above or below tight bounds and remained the only one for several years, at least for constraint satisfaction problems (CSPs). However, in the last few years the study of CSPs parameterized above or below tight bounds has finally picked up. This is, in large part, due to emergence of new probabilistic and linear-algebraic methods and approaches in the area.

In this survey paper, we will overview several results on CSPs parameterized above or below tight bounds, as well as some methods used to obtain these results. While not going into details of the proofs, we will discuss some ideas behind the proofs. We will also consider several open problems in the area.

In the remainder of this section we give a brief overview of the paper and its organization.

In the next section we provide basics on parameterized algorithms and complexity. The notions mentioned there are all well-known apart from a recent notion of a bikernel introduced by Alon et al. [2]. In Section 3, we describe some probabilistic and Harmonic Analysis tools. These tools are, in particular, used in the recently introduced Strictly-Above-Below-Expectation method [30].

Results on MAXSAT parameterized above or below tight bounds are discussed in Section 4. We will consider the above-mentioned parameterization of MAXSAT above tight bound, some “stronger” parameterizations of MAXSAT introduced or inspired by Mahajan and Raman [44]. The stronger parameterizations are based on the notion of a t -satisfiable CNF formula (a formula in which each set of t clauses can be satisfied by a truth assignment) and asymptotically tight lower bounds on the maximum number of clauses of a t -satisfiable CNF formula

satisfied by a truth assignment for $t = 2$ and 3. We will describe linear-variable kernels obtained for both $t = 2$ and 3.

We will also consider the parameterization of 2-SAT below the upper bound m , the number of clauses. This problem was proved to be fixed-parameter tractable by Razgon and O’Sullivan [52]. Raman et al. [51] and Cygan et al. [17] designed faster parameterized algorithms for the problem. The problem has several applications, which we will briefly overview.

Boolean Maximum r -CSPs parameterized above the average value are considered in Section 5, where r is a positive integral constant. In general, the Maximum r -CSP is given by a set V of n variables and a set of m Boolean formulas; each formula is assigned an integral positive weight and contains at most r variables from V . The aim is to find a truth assignment which maximizes the weight of satisfied formulas. Averaging over all truth assignments, we can find the average value A of the weight of satisfied formulas. It is easy to show that we can always find a truth assignment to the variables of V which satisfied formulas of total weight at least A . Thus, a natural parameterized problem is whether there exists a truth assignment that satisfies formulas of total weight at least $A + k$, where k is the parameter (k is a nonnegative integer). We denote such a problem by MAX- r -CSP-AA.

The problem MAX- r -LIN2-AA is a special case of MAX- r -CSP-AA when every formula is a linear equation over \mathbb{F}_2 with at most r variables. For MAX- r -LIN2-AA, we have $A = W/2$, where W is the total weight of all equations. It is well-known that, in polynomial time, we can find an assignment to the variables that satisfies equations of total weight at least $W/2$, but, for any $\epsilon > 0$ it is NP-hard to decide whether there is an assignment satisfying equations of total weight at least $W(1 + \epsilon)/2$ [33]. We give proof schemes of a result by Gutin et al. [30] that MAX- r -LIN2-AA has a kernel of quadratic size and a result of Crowston, et al. [12] that MAX- r -LIN2-AA has a kernel with at most $(2k - 1)r$ variables. The latest result improves that of Kim and Williams [39] that MAX- r -LIN2-AA has a kernel with at most $r(r + 1)k$ variables. Papers [12,39] imply an algorithm of runtime $2^{O(k)} + m^{O(1)}$ for MAX- r -LIN2-AA.

We give a proof scheme of a result by Alon et al. [2] that MAX- r -CSP-AA has a kernel of polynomial size. The main idea of the proof is to reduce MAX- r -CSP-AA to MAX- r -LIN2-AA and use results on MAX- r -LIN2-AA and a lemma on bikernels given in the next section. The result of Alon et al. [2] solves an open question of Mahajan, Raman and Sikdar [45] not only for MAX- r -SAT-AA but for the more general problem MAX- r -CSP-AA. The problem MAX- r -SAT-AA is a special case of MAX- r -CSP-AA when every formula is a clause with at most r variables. For MAX- r -SAT-AA, the reduction to MAX- r -LIN2-AA can be complemented by a reduction from MAX- r -LIN2-AA back to MAX- r -SAT-AA, which yields a kernel of quadratic size. (Note that while the size of the kernel for MAX- r -CSP-AA is polynomial we are unable to bound the degree of the polynomial.)

MAXLIN2-AA is the same problem as MAX- r -LIN2-AA, but the number of variables in an equation is not bounded. Thus, MAXLIN2-AA is a generalization

of MAX- r -LIN2-AA. Section 6 presents a scheme of a recent proof by Crowston, Fellows et al. [12] that MAXLIN2-AA is fixed-parameter tractable and has a kernel with polynomial number of variables. This result finally solved an open question of Mahajan, Raman and Sikdar [45]. Still, we do not know whether MAXLIN2-AA has a kernel of polynomial size and we present only partial results on the topic. MAX-SAT-AA is the same problem as MAX- r -SAT-AA, but the number of variables in a clause is not bounded. Crowston et al. [15] proved that MAX-SAT-AA is para-NP-complete and, thus, MAXSAT-AA is not fixed-parameter tractable unless $P=NP$. We give a short discussion of this result in the end of Section 6.

In Section 7 we discuss parameterizations above average of Ordering CSPs of arities 2 and 3. It turns out that for our parameterization the most important Ordering CSP is the problem r -LINEAR ORDERING ($r \geq 2$). An instance of r -LINEAR ORDERING consists of a set V of variables and a multiset C of constraints, which are ordered r -tuples of distinct variables of V (note that the same set of r variables may appear in several different constraints). The objective is to find an ordering α of V that maximizes the number of constraints whose order in α follows that of the constraint (such constraints are *satisfied* by α).

It is easy to see that $|C|/r!$ is the average number of constraints satisfied by an ordering of V and that it is a tight lower bound on the maximum number of constraints satisfied by an ordering of V . The only nontrivial Ordering CSP of arity 2 is 2-LINEAR ORDERING. For this problem, Guruswami, Manokaran and Raghavendra [26] proved that it is impossible to find, in polynomial time, an ordering that satisfies at least $|C|(1 + \epsilon)/2$ constraints for every $\epsilon > 0$ provided the Unique Games Conjecture (UGC) of Khot [38] holds. Similar approximation resistant results were proved for all Ordering CSPs of arity 3 by Charikar, Guruswami and Manokaran [8] and for Ordering CSPs of any arity by Guruswami et al. [25].

In the problem r -LINEAR ORDERING parameterized above average (r -LINEAR ORDERING-AA), given an instance of r -LINEAR ORDERING with a multiset C of constraints, we are to decide whether there is an ordering satisfying at least $|C|/r! + k$ constraints, where k is the parameter. Gutin et al. [30] proved that 2-LINEAR ORDERING-AA is fixed-parameter tractable and, moreover, has a kernel of a quadratic size. BETWEENNESS is an Ordering CSP of arity 3, which is formulated in Section 7. Gutin et al. [29] solved an open question of Benny Chor stated in Niedermeier's monograph [48] by showing that BETWEENNESS parameterized above average is fixed-parameter tractable and, moreover, has a kernel of a quadratic size.

A simple, yet important, observation is that all Ordering CSPs of arity 3 parameterized above average can be reduced, in polynomial time, to 3-LINEAR ORDERING parameterized above average (3-LINEAR ORDERING-AA) and that this reduction preserves the parameter. Thus, to prove that all Ordering CSPs of arity 3 parameterized above average are fixed-parameter tractable, it suffices to show that 3-LINEAR ORDERING-AA is fixed-parameter tractable.

Gutin et al. [27] proved that 3-LINEAR ORDERING-AA is fixed-parameter tractable and, moreover, has a kernel with a quadratic number of constraints and variables.

Kim and Williams [39] partially improved the results above by showing that 2-LINEAR ORDERING-AA and 3-LINEAR ORDERING-AA have kernels with linear number of variables. Parameterized complexity of Ordering CSPs of arities 4 and higher is still unknown. It seems to be technically much more difficult to prove that 4-LINEAR ORDERING-AA is fixed-parameter tractable than that 3-LINEAR ORDERING-AA is fixed-parameter tractable.

2 Basics on Parameterized Algorithms and Complexity

A parameterized problem Π can be considered as a set of pairs (I, k) where I is the *problem instance* and k (usually a nonnegative integer) is the *parameter*. Π is called *fixed-parameter tractable (fpt)* if membership of (I, k) in Π can be decided by an algorithm of runtime $O(f(k)|I|^c)$, where $|I|$ is the size of I , $f(k)$ is an arbitrary function of the parameter k only, and c is a constant independent from k and I . Such an algorithm is called an *fpt algorithm*. Let Π and Π' be parameterized problems with parameters k and k' , respectively. An *fpt-reduction R from Π to Π'* is a many-to-one transformation from Π to Π' , such that (i) $(I, k) \in \Pi$ if and only if $(I', k') \in \Pi'$ with $k' \leq g(k)$ for a fixed computable function g , and (ii) R is of complexity $O(f(k)|I|^c)$.

If the nonparameterized version of Π (where k is just part of the input) is NP-hard, then the function $f(k)$ must be superpolynomial provided $P \neq NP$. Often $f(k)$ is “moderately exponential,” which makes the problem practically tractable for small values of k . Thus, it is important to parameterize a problem in such a way that the instances with small values of k are of real interest.

When the decision time is replaced by the much more powerful $|I|^{O(f(k))}$, we obtain the class XP, where each problem is polynomial-time solvable for any fixed value of k . There is an infinite number of parameterized complexity classes between FPT and XP (for each integer $t \geq 1$, there is a class $W[t]$) and they form the following tower:

$$FPT \subseteq W[1] \subseteq W[2] \subseteq \dots \subseteq W[P] \subseteq XP.$$

Here $W[P]$ is the class of all parameterized problems (I, k) that can be decided in $f(k)|I|^{O(1)}$ time by a nondeterministic Turing machine that makes at most $f(k) \log |I|$ nondeterministic steps for some function f . For the definition of classes $W[t]$, see, e.g., [22] (we do not use these classes in the rest of the paper).

Π is in *para-NP* if membership of (I, k) in Π can be decided in nondeterministic time $O(f(k)|I|^c)$, where $|I|$ is the size of I , $f(k)$ is an arbitrary function of the parameter k only, and c is a constant independent from k and I . Here, nondeterministic time means that we can use nondeterministic Turing machine. A parameterized problem Π' is *para-NP-complete* if it is in para-NP and for any parameterized problem Π in para-NP there is an fpt-reduction from Π to Π' .

While several fpt algorithms were designed many years ago (e.g., pseudo-polynomial algorithms with parameter being the binary length of the maximum number, cf. [23]), Downey and Fellows were the first to systematically study the theory of parameterized algorithms and complexity and they wrote the first monograph [18] in the area².

Given a pair Π, Π' of parameterized problems, a *bikernelization from Π to Π'* is a polynomial-time algorithm that maps an instance (I, k) to an instance (I', k') (the *bikernel*) such that (i) $(I, k) \in \Pi$ if and only if $(I', k') \in \Pi'$, (ii) $k' \leq f(k)$, and (iii) $|I'| \leq g(k)$ for some functions f and g . The function $g(k)$ is called the *size* of the bikernel. A *kernelization* of a parameterized problem Π is simply a bikernelization from Π to itself. Then (I', k') is a *kernel*. The term bikernel was coined by Alon et al. [2]; in [5] a bikernel is called a generalized kernel.

It is well-known that a parameterized problem Π is fixed-parameter tractable if and only if it is decidable and admits a kernelization [18,22,48]. This result can be extended as follows: A decidable parameterized problem Π is fixed-parameter tractable if and only if it admits a bikernelization from itself to a decidable parameterized problem Π' [2].

Due to applications, low degree polynomial size kernels are of main interest. Unfortunately, many fixed-parameter tractable problems do not have kernels of polynomial size unless the polynomial hierarchy collapses to the third level [5,6,20]. For further background and terminology on parameterized complexity we refer the reader to the monographs [18,22,48].

The following lemma of Alon et al. [2] inspired by a lemma from [6] shows that polynomial bikernels imply polynomial kernels.

Lemma 1. *Let Π, Π' be a pair of decidable parameterized problems such that the nonparameterized version of Π' is in NP, and the nonparameterized version of Π is NP-complete. If there is a bikernelization from Π to Π' producing a bikernel of polynomial size, then Π has a polynomial-size kernel.*

Henceforth $[n]$ stands for the set $\{1, 2, \dots, n\}$.

3 Probabilistic and Harmonic Analysis Tools

We start this section by outlining the very basic principles of the probabilistic method which will be implicitly used in this paper. Given random variables X_1, \dots, X_n , the fundamental property known as *linearity of expectation* states that $\mathbb{E}(X_1 + \dots + X_n) = \mathbb{E}(X_1) + \dots + \mathbb{E}(X_n)$. The *averaging argument* utilizes the fact that there is a point for which $X \geq \mathbb{E}(X)$ and a point for which $X \leq \mathbb{E}(X)$ in the probability space. Also a positive probability $\mathbb{P}(A) > 0$ for some event A means that there is at least one point in the probability space which belongs to A . For example, $\mathbb{P}(X \geq k) > 0$ tells us that there exists a point for which $X \geq k$.

A random variable is *discrete* if its distribution function has a finite or countable number of positive increases. A random variable X is *symmetric* if $-X$ has

² Michael R. Fellows has worked tirelessly for many years to promote the area and so can be affectionately called St. Paul of Parameterized Complexity.

the same distribution function as X . If X is discrete, then X is symmetric if and only if $\mathbb{P}(X = a) = \mathbb{P}(X = -a)$ for each real a . Let X be a symmetric variable for which the first moment $\mathbb{E}(X)$ exists. Then $\mathbb{E}(X) = \mathbb{E}(-X) = -\mathbb{E}(X)$ and, thus, $\mathbb{E}(X) = 0$. The following is easy to prove [30].

Lemma 2. *If X is a symmetric random variable and $\mathbb{E}(X^2)$ is finite, then*

$$\mathbb{P}(X \geq \sqrt{\mathbb{E}(X^2)}) > 0.$$

If X is not symmetric then the following lemma can be used instead (a similar result was already proved in [3]).

Lemma 3 (Alon et al. [2]). *Let X be a real random variable and suppose that its first, second and fourth moments satisfy $\mathbb{E}[X] = 0$, $\mathbb{E}[X^2] = \sigma^2 > 0$ and $\mathbb{E}[X^4] \leq c\mathbb{E}[X^2]^2$, respectively, for some constant c . Then $\mathbb{P}(X > \frac{\sigma}{\sqrt{c}}) > 0$.*

To check $\mathbb{E}[X^4] \leq c\mathbb{E}[X^2]^2$ we often can use the following well-known inequality.

Lemma 4 (Hypercontractive Inequality [7]). *Let $f = f(x_1, \dots, x_n)$ be a polynomial of degree r in n variables x_1, \dots, x_n each with domain $\{-1, 1\}$. Define a random variable X by choosing a vector $(\epsilon_1, \dots, \epsilon_n) \in \{-1, 1\}^n$ uniformly at random and setting $X = f(\epsilon_1, \dots, \epsilon_n)$. Then $\mathbb{E}[X^4] \leq 9^r \mathbb{E}[X^2]^2$.*

If $f = f(x_1, \dots, x_n)$ is a polynomial in n variables x_1, \dots, x_n each with domain $\{-1, 1\}$, then it can be written as $f = \sum_{I \subseteq [n]} c_I \prod_{i \in S} x_i$, where $[n] = \{1, \dots, n\}$ and c_I is a real for each $I \subseteq [n]$.

The following dual, in a sense, form of the Hypercontractive Inequality was proved by Gutin and Yeo [31]; for a weaker result, see [30].

Lemma 5. *Let $f = f(x_1, \dots, x_n)$ be a polynomial in n variables x_1, \dots, x_n each with domain $\{-1, 1\}$ such that $f = \sum_{I \subseteq [n]} c_I \prod_{i \in S} x_i$. Suppose that no variable x_i appears in more than ρ monomials of f . Define a random variable X by choosing a vector $(\epsilon_1, \dots, \epsilon_n) \in \{-1, 1\}^n$ uniformly at random and setting $X = f(\epsilon_1, \dots, \epsilon_n)$. Then $\mathbb{E}[X^4] \leq (2\rho + 1)\mathbb{E}[X^2]^2$.*

The following lemma is easy to prove, cf. [30]. In fact, the equality there is a special case of Parseval’s Identity in Harmonic Analysis, cf. [49].

Lemma 6. *Let $f = f(x_1, \dots, x_n)$ be a polynomial in n variables x_1, \dots, x_n each with domain $\{-1, 1\}$ such that $f = \sum_{I \subseteq [n]} c_I \prod_{i \in I} x_i$. Define a random variable X by choosing a vector $(\epsilon_1, \dots, \epsilon_n) \in \{-1, 1\}^n$ uniformly at random and setting $X = f(\epsilon_1, \dots, \epsilon_n)$. Then $\mathbb{E}[X^2] = \sum_{i \in I} c_I^2$.*

4 Parameterizations of MaxSat

In the well-known problem MAXSAT, we are given a CNF formula F with m clauses and asked to determine the maximum number of clauses of F that can be satisfied by a truth assignment. Let us assign TRUE to each variable of F with

probability $1/2$ and observe that the probability of a clause to be satisfied is at least $1/2$ and thus, by linearity of expectation, the expected number of satisfied clauses in F is at least $m/2$. Thus, by the averaging argument, there exists a truth assignment to the variables of F which satisfies at least $m/2$ clauses of F .

Let us denote by $\text{sat}(F)$ the maximum number of clauses of F that can be satisfied by a truth assignment. The lower bound $\text{sat}(F) \geq m/2$ is tight as we have $\text{sat}(H) = m/2$ if $H = (x_1) \wedge (\bar{x}_1) \wedge \dots \wedge (x_{m/2}) \wedge (\bar{x}_{m/2})$. Consider the following parameterization of MAXSAT above tight lower bound introduced by Mahajan and Raman [44].

MAXSAT-A($m/2$)

Instance: A CNF formula F with m clauses (clauses may appear several times in F) and a nonnegative integer k .

Parameter: k .

Question: $\text{sat}(F) \geq m/2 + k$?

Mahajan and Raman [44] proved that MAXSAT-A($m/2$) admits a kernel with at most $6k + 3$ variables and $10k$ clauses. Crowston et al. [16] improved this result, by obtaining a kernel with at most $4k$ variables and $(2\sqrt{5} + 4)k$ clauses. The improved result is a simple corollary of a new lower bound on $\text{sat}(F)$ obtained in [16], which is significantly stronger than the simple bound $\text{sat}(F) \geq m/2$. We give the new lower bound below, in Theorem 3.

For a variable x in F , let $m(x)$ denote the number of pairs of unit clauses $(x), (\bar{x})$ that have to be deleted from F such that F has no pair $(x), (\bar{x})$ any longer. Let $\text{var}(F)$ be the set of all variables in F and let $\ddot{m} = \sum_{x \in \text{var}(F)} m(x)$. The following is a stronger lower bound on $\text{sat}(F)$ than $m/2$.

Theorem 1. *For a CNF formula F , we have $\text{sat}(F) \geq \ddot{m}/2 + \hat{\phi}(m - \ddot{m})$, where $\hat{\phi} = (\sqrt{5} - 1)/2 \approx 0.618$.*

A CNF formula F is t -satisfiable if for any t clauses in F , there is a truth assignment which satisfies all of them. It is easy to check that F is 2-satisfiable if and only if $\ddot{m} = 0$ and clearly Theorem 1 is equivalent to the assertion that if F is 2-satisfiable then $\text{sat}(F) \geq \hat{\phi}m$. The proof of this assertion by Lieberherr and Specker [41] is quite long; Yannakakis [56] gave the following short probabilistic proof. For $x \in \text{var}(F)$, let the probability of x being assigned TRUE be $\hat{\phi}$ if (x) is in F , $1 - \hat{\phi}$ if (\bar{x}) is in F , and $1/2$, otherwise, independently of the other variables. Let us bound the probability $p(C)$ of a clause C to be satisfied. If C contains only one literal, then, by the assignment above, $p(C) = \hat{\phi}$. If C contains two literals, then, without loss of generality, $C = (x \vee y)$. Observe that the probability of x assigned FALSE is at most $\hat{\phi}$ (it is $\hat{\phi}$ if (\bar{x}) is in F). Thus, $p(C) \geq 1 - \hat{\phi}^2$. It remains to observe that $1 - \hat{\phi}^2 = \hat{\phi}$. Now to obtain the bound $\text{sat}(F) \geq \hat{\phi}m$ apply linearity of expectation and the averaging argument.

Note that $\hat{\phi}m$ is an asymptotically tight lower bound: for each $\epsilon > 0$ there are 2-satisfiable CNF formulae F with $\text{sat}(F) < m(\hat{\phi} + \epsilon)$ [41]. Thus, the following problem stated by Mahajan and Raman [44] is natural.

MAX-2S-SAT-A($\hat{\phi}m$)

Instance: A 2-satisfiable CNF formula F with m clauses (clauses may appear several times in F) and a nonnegative integer k .

Parameter: k .

Question: $\text{sat}(F) \geq \hat{\phi}m + k$?

Mahajan and Raman [44] conjectured that MAX-2S-SAT-A($\hat{\phi}m$) is fpt. Crowston et al. [16] solved this conjecture in the affirmative; moreover, they obtained a kernel with at most $(7 + 3\sqrt{5})k$ variables. This result is an easy corollary from a lower bound on $\text{sat}(F)$ given in Theorem 3, which, for 2-satisfiable CNF formulas, is stronger than the one in Theorem 1. The main idea of [16] is to obtain a lower bound on $\text{sat}(F)$ that includes the number of variables as a factor. It is clear that for general CNF formula F such a bound is impossible. For consider a formula containing a single clause c containing a large number of variables. We can arbitrarily increase the number of variables in the formula, and the maximum number of satisfiable clauses will always be 1. We therefore need a reduction rule that cuts out ‘excess’ variables. Our reduction rule is based on the notion of an expanding formula given below. Lemma 7 and Theorem 2 show the usefulness of this notion.

A CNF formula F is called *expanding* if for each $X \subseteq \text{var}(F)$, the number of clauses containing at least one variable from X is at least $|X|$ [21,55]. The following lemma and its parts were proved by many authors, see, e.g., Fleischner, Kullmann and Szeider [21], Lokshtanov [43] and Szeider [55].

Lemma 7. *Let F be a CNF formula and let V and C be its sets of variables and clauses. There exists a subset $C^* \subseteq C$ that can be found in polynomial time, such that the formula F' with clauses $C \setminus C^*$ and variables $V \setminus V^*$, where $V^* = \text{var}(C^*)$, is expanding. Moreover, $\text{sat}(F) = \text{sat}(F') + |C^*|$.*

The following result was shown by Crowston et al. [16]. The proof is nontrivial and consists of a deterministic algorithm for finding the corresponding truth assignment and a detailed combinatorial analysis of the algorithm.

Theorem 2. *Let F be an expanding 2-satisfiable CNF formula with n variables and m clauses. Then $\text{sat}(F) \geq \hat{\phi}m + n(2 - 3\hat{\phi})/2$.*

Lemma 7 and Theorem 2 imply the following:

Theorem 3. *Let F be a 2-satisfiable CNF formula and let V and C be its sets of variables and clauses. There exists a subset $C^* \subseteq C$ that can be found in polynomial time, such that the formula F' with clauses $C \setminus C^*$ and variables $V \setminus V^*$, where $V^* = \text{var}(C^*)$, is expanding. Moreover, we have*

$$\text{sat}(F) \geq \hat{\phi}m + (1 - \hat{\phi})m^* + (n - n^*)(2 - 3\hat{\phi})/2,$$

where $m = |C|$, $m^* = |C^*|$, $n = |V|$ and $n^* = |V^*|$.

Let us turn now to 3-satisfiable CNF formulas. If F is 3-satisfiable then it is not hard to check that the forbidden sets of clauses are pairs of the form $\{x\}, \{\bar{x}\}$ and triplets of the form $\{x\}, \{y\}, \{\bar{x}, \bar{y}\}$ or $\{x\}, \{\bar{x}, y\}, \{\bar{x}, \bar{y}\}$, as well as any triplets that can be derived from these by switching positive literals with negative literals.

Lieberherr and Specker [42] and, later, Yannakakis [56] proved the following: if F is 3-satisfiable then $\text{sat}(F) \geq \frac{2}{3}w(\mathcal{C}(F))$. This bound is also asymptotically tight. Yannakakis [56] gave a probabilistic proof which is similar to his proof for 2-satisfiable formulas, but requires consideration of several cases and, thus, not as short as for 2-satisfiable formulas. For details of his proof, see, e.g., Gutin, Jones and Yeo [28] and Jukna [36] (Theorem 20.6). Yannakakis's approach was extended by Gutin, Jones and Yeo [28] to prove the following theorem using a quite complicated probabilistic distribution for a random truth assignment.

Theorem 4. *Let F be an expanding 3-satisfiable CNF formula with n variables and m clauses. Then $\text{sat}(F) \geq \frac{2}{3}m + \rho n$, where $\rho(> 0.0019)$ is a constant.*

This theorem and Lemma 7 imply the following:

Theorem 5. *Let F be a 3-satisfiable CNF formula and let V and C be its sets of variables and clauses. There exists a subset $C^* \subseteq C$ that can be found in polynomial time, such that the formula F' with clauses $C \setminus C^*$ and variables $V \setminus V^*$, where $V^* = \text{var}(C^*)$, is expanding. Moreover, we have*

$$\text{sat}(F) \geq \frac{2}{3}m + \frac{1}{3}m^* + \rho(n - n^*),$$

where $\rho(> 0.0019)$ is a constant, $m = |C|$, $m^* = |C^*|$, $n = |V|$ and $n^* = |V^*|$.

Using this theorem it is easy to obtain a linear-in-number-of-variables kernel for the following natural analog of MAX-2S-SAT-A($\hat{\phi}m$), see [28] for details.

MAX-3S-SAT-A($\frac{2}{3}m$)

Instance: A 3-satisfiable CNF formula F with m clauses and a nonnegative integer k .

Parameter: k .

Question: $\text{sat}(F) \geq \frac{2}{3}m + k$?

Now let us consider the following important parameterization of r -SAT below the tight upper bound m :

r -SAT-B(m)

Instance: An r -CNF formula F with m clauses (every clause has at most r literals) and a nonnegative integer k .

Parameter: k .

Question: $\text{sat}(F) \geq m - k$?

Since MAX- r -SAT is NP-hard for each fixed $r \geq 3$, r -SAT-B(m) is not fpt unless P=NP. However, the situation changes for $r = 2$: Razgon and O’Sullivan [52] proved that 2-SAT-B(m) is fpt. The algorithm in [52] is of complexity $O(15^k km^3)$ and, thus, MAX-2-SAT-B(m) admits a kernel with at most $15^k k$ clauses. It is not known whether 2-SAT-B(m) admits a kernel with a polynomial number of variables. Raman et al. [51] and Cygan et al. [17] designed algorithms for 2-SAT-B(m) of runtime $9^k (km)^{O(1)}$ and $4^k (km)^{O(1)}$, respectively. In both papers, the authors consider the following parameterized problem (VC-AMM): given a graph G whose maximum matching is of cardinality μ , decide whether G has a vertex cover with at most $\mu + k$ vertices, where k is the parameter. A parameterized algorithm of the above-mentioned complexity actually is obtained for VC-AMM, and 2-SAT-B(m) is polynomially transformed into VC-AMM (the transformation is parameter-preserving). While Raman et al. [51] obtain the parameterized algorithm for VC-AMM directly, Cygan et al. [17] derive it via a reduction from a more general problem on graphs parameterized above a tight bound.

2-SAT-B(m) has several application. 2-SAT-B(m) is, in fact, equivalent to VC-AMM [46,51,17]. Mishra et al. [46] studied the following problem: given a graph G , decide whether by deleting at most k vertices we can make G *König*, i.e., a graph in which the minimum size of a vertex cover equals the maximum number of edges in a matching. They showed how to reduce the last problem to VC-AMM. It is noted by Gottlob and Szeider [24] that fixed-parameter tractability of VC-AMM implies the fixed-parameter tractability of the following problem. Given a CNF formula F (not necessarily 2-CNF), decide whether there exists a subset of at most k variables of F so that after removing all occurrences of these variables from the clauses of F , the resulting CNF formula is *Renamable Horn*, i.e., it can be transformed by renaming of the variables into a CNF formula with at most one positive literal in each clause.

2-SAT-B(m) has also been used in order to obtain the best known bound on the order of a kernel for VERTEX COVER (given a graph G and an integer k , decide whether G has a vertex cover with at most k vertices). The fact that VERTEX COVER has a kernel with at most $2k$ vertices was known for a long time, see Chen, Kanj and Jia [9]. This was improved to $2k - 1$ by Chlebík and Clebíkóvá [10] and further to $2k - c$ for any constant c by Soleimanfallah and Yeó [54]. Lampis [40] used the same approach as in [54], but instead of reducing an instance of VERTEX COVER to a large number of 2-SAT instances, he reduced VERTEX COVER to 2-SAT-B(m) via VC-AMM. As a result, Lampis [40] obtained a kernel of order at most $2k - c \log k$ for any constant c . We will now briefly describe how this kernel was obtained.

For a graph G let $\beta(G)$ denotes the minimum size of a vertex cover of G and $\mu(G)$ the maximum size of a matching in G . In their classical work Nemhauser and Trotter [47] proved the following:

Theorem 6. *There is an $O(|E|\sqrt{|V|})$ -time algorithm which for a given graph $G = (V, E)$ computes two disjoint subsets of vertices of G , V' , V'' , such that $\beta(G) = \beta(G[V']) + |V''|$ and $\beta(G[V']) \geq |V'|/2$.*

Soleimanfallah and Yeo [54] showed the following additional inequality:

$$\beta(G[V']) \geq |V'| - \mu(G). \tag{1}$$

Let $k' := k - |V''|$. By Theorem 6, $\beta(G) \leq k$ if and only if $\beta(G[V']) \leq k'$. If $|V'| \leq 2k' - c \log k' \leq 2k - c \log k$ then we have a kernel and we are done. Thus, it suffices to show that if $|V'| > 2k' - c \log k'$ we can decide whether $\beta(G[V']) \leq k'$ in polynomial time. We assume that $|V'| > 2k' - c \log k'$ and we may also assume that $|V'| \leq 2k'$ as otherwise $\beta(G[V']) > k'$ by Theorem 6. By (1) if $\mu(G[V']) \leq (|V'| - c \log k')/2$ then $\beta(G[V']) \geq (|V'| + c \log k')/2$. Since $|V'| > 2k' - c \log k'$ this means that $\beta(G[V']) > k'$.

So, consider the case $\mu(G[V']) > (|V'| - c \log k')/2$. Since $|V'| > 2k' - c \log k'$ and $\mu(G[V']) > (|V'| - c \log k')/2$, we have $\mu(G[V']) > k' - c \log k'$ and so $k' < \mu(G[V']) + c \log k'$. Thus, to decide whether $\beta(G[V']) \leq k'$ it suffices to compute ℓ such that $\beta(G[V']) = \mu(G[V']) + \ell$, where $\ell < c \log k'$, and to compare $\mu(G[V']) + \ell$ with k' . Using an fpt algorithm for VC-AMM (which is essentially an fpt algorithm for MAX-2-SAT-B(m) as the two problems are equivalent) we can compute ℓ in fpt time (provided we use an efficient algorithm such as in [52,51,17]).

5 Boolean Max- r -CSPs above Average

Throughout this section, r is a positive integral constant. Recall that the problem MAX- r -CSP-AA is given by a set V of n variables and a set of m Boolean formulas; each formula is assigned an integral positive weight and contains at most r variables from V . Averaging over all truth assignments, we can find the average value A of the weight of satisfied formulas. We wish to decide whether there exists a truth assignment that satisfies formulas of total weight at least $A + k$, where k is the parameter (k is a nonnegative integer).

Recall that the problem MAX- r -LIN2-AA is a special case of MAX- r -CSP-AA when every formula is a linear equation over \mathbb{F}_2 with at most r variables and that MAX-LIN2-AA is the extension of MAX- r -LIN2-AA when we do not bound the number of variables in an equation. Research of both MAX- r -LIN2-AA and MAX-LIN2-AA led to a number of basic notions and results of interest for both problems, and we devote Subsection 5.1 to these notions and results. In particular, we will show that $A = W/2$, where W is the total weight of all equations, introduce a Gaussian-elimination-type algorithm for both problems, and a notion and simple lemma of a sum-free subset of a set of vectors in \mathbb{F}_2^n . This lemma is a key ingredient in proving some important results for MAX- r -LIN2-AA and MAX-LIN2-AA.

MAX- r -LIN2-AA is studied in Subsection 5.2, where we give proof schemes of a result by Gutin et al. [30] that MAX- r -LIN2-AA has a kernel of quadratic size and a result of Crowston, Fellows et al. [12] that MAX- r -LIN2-AA has a kernel with at most $(2k - 1)r$ variables. The latest result improves that of Kim and Williams [39] that MAX- r -LIN2-AA has a kernel with at most $r(r + 1)k$ variables.

In Subsection 5.3, we give a proof scheme of a result by Alon et al. [2] that MAX- r -CSP-AA has a kernel of polynomial size. The main idea of the proof is to reduce MAX- r -CSP-AA to MAX- r -LIN2-AA and use the above results on MAX- r -LIN2-AA and Lemma 1. This shows the existence of a polynomial-size kernel, but does not allow us to obtain a bound on the degree of the polynomial. Nevertheless, this solves an open question of Mahajan, Raman and Sikdar [45] not only for MAX- r -SAT-AA but also for the more general problem MAX- r -CSP-AA. Recall that the problem MAX- r -SAT-AA is a special case of MAX- r -CSP-AA when every formula is a clause with at most r variables. For MAX- r -SAT-AA, the reduction to MAX- r -LIN2-AA can be complemented by a reduction from MAX- r -LIN2-AA back to MAX- r -SAT-AA, which yields a kernel of quadratic size.

5.1 Basic Results for Max-LIN2-AA and Max- r -Lin2-AA

Recall that in the problems MAXLIN2-AA and MAX- r -LIN2-AA, we are given a system S consisting of m linear equations in n variables over \mathbb{F}_2 in which each equation is assigned a positive integral weight. In MAX- r -LIN2-AA, we have an extra constraint that every equation has at most r variables. Let us write the system S as $\sum_{i \in I} z_i = b_I$, $I \in \mathcal{F}$, and let w_I denote the weight of an equation $\sum_{i \in I} z_i = b_I$. Clearly, $m = |\mathcal{F}|$. Let $W = \sum_{I \in \mathcal{F}} w_I$ and let $\text{sat}(S)$ be the maximum total weight of equations that can be satisfied simultaneously.

For each $i \in [n]$, set $z_i = 1$ with probability $1/2$ independently of the rest of the variables. Then each equation is satisfied with probability $1/2$ and the expected weight of satisfied equations is $W/2$ (as our probability distribution is uniform, $W/2$ is also the average weight of satisfied equations). Hence $W/2$ is a lower bound; to see its tightness consider a system of pairs of equations of the form $\sum_{i \in I} z_i = 0$, $\sum_{i \in I} z_i = 1$ of weight 1. The aim in both MAX-LIN2-AA and MAX- r -LIN2-AA is to decide whether for the given system S , $\text{sat}(S) \geq W/2 + k$, where k is the parameter. It is well-known that, in polynomial time, we can find an assignment to the variables that satisfies equations of total weight at least $W/2$, but, for any $\epsilon > 0$ it is NP-hard to decide whether there is an assignment satisfying equations of total weight at least $W(1 + \epsilon)/2$ [33].

Henceforth, it will often be convenient for us to consider linear equations in their multiplicative form, i.e., instead of an equation $\sum_{i \in I} z_i = b_I$ with $z_i \in \{0, 1\}$, we will consider the equation $\prod_{i \in I} x_i = (-1)^{b_I}$ with $x_i \in \{-1, 1\}$. Clearly, an assignment $z^0 = (z_1^0, \dots, z_n^0)$ satisfies $\sum_{i \in I} z_i = b_I$ if and only if the assignment $x^0 = (x_1^0, \dots, x_n^0)$ satisfies $\prod_{i \in I} x_i = (-1)^{b_I}$, where $x_i^0 = (-1)^{z_i^0}$ for each $i \in [n]$.

Let $\varepsilon(x) = \sum_{I \in \mathcal{F}} w_I (-1)^{b_I} \prod_{i \in I} x_i$ (each $x_i \in \{-1, 1\}$) and note that $\varepsilon(x^0)$ is the difference between the total weight of satisfied and falsified equations when $x_i = x_i^0$ for each $i \in [n]$. Crowston et al. [14] call $\varepsilon(x)$ the *excess* and the maximum possible value of $\varepsilon(x)$ the *maximum excess*.

Remark 1. Observe that the answer to MAX-LIN2-AA and MAX- r -LIN2-AA is YES if and only if the maximum excess is at least $2k$.

Let A be the matrix over \mathbb{F}_2 corresponding to the set of equations in S , such that $a_{ji} = 1$ if $i \in I_j$ and 0, otherwise.

Consider two reduction rules for MAX-LIN2-AA introduced by Gutin et al. [30]. Rule 1 was studied before in [34].

Reduction Rule 1. *If we have, for a subset I of $[n]$, an equation $\prod_{i \in I} x_i = b'_I$ with weight w'_I , and an equation $\prod_{i \in I} x_i = b''_I$ with weight w''_I , then we replace this pair by one of these equations with weight $w'_I + w''_I$ if $b'_I = b''_I$ and, otherwise, by the equation whose weight is bigger, modifying its new weight to be the difference of the two old ones. If the resulting weight is 0, we delete the equation from the system.*

Reduction Rule 2. *Let $t = \text{rank}A$ and suppose columns a^{i_1}, \dots, a^{i_t} of A are linearly independent. Then delete all variables not in $\{x_{i_1}, \dots, x_{i_t}\}$ from the equations of S .*

Lemma 8. [30] *Let S' be obtained from S by Rule 1 or 2. Then the maximum excess of S' is equal to the maximum excess of S . Moreover, S' can be obtained from S in time polynomial in n and m .*

If we cannot change a weighted system S using Rules 1 and 2, we call it *irreducible*.

Let S be an irreducible system of MAX-LIN2-AA. Consider the following algorithm introduced in [14]. We assume that, in the beginning, no equation or variable in S is marked.

ALGORITHM \mathcal{H}

While the system S is nonempty do the following:

1. Choose an equation $\prod_{i \in I} x_i = b$ and mark a variable x_l such that $l \in I$.
2. Mark this equation and delete it from the system.
3. Replace every equation $\prod_{i \in I'} x_i = b'$ in the system containing x_l by $\prod_{i \in I \Delta I'} x_i = bb'$, where $I \Delta I'$ is the symmetric difference of I and I' (the weight of the equation is unchanged).
4. Apply Reduction Rule 1 to the system.

The *maximum \mathcal{H} -excess* of S is the maximum possible total weight of equations marked by \mathcal{H} for S taken over all possible choices in Step 1 of \mathcal{H} . The following lemma indicates the potential power of \mathcal{H} .

Lemma 9. [14] *Let S be an irreducible system. Then the maximum excess of S equals its maximum \mathcal{H} -excess.*

This lemma gives no indication on how to choose equations in Step 1 of Algorithm \mathcal{H} . As the problem MAX-LIN2-AA is NP-hard, we cannot hope to obtain an polynomial-time procedure for optimal choice of equations in Step 1 and, thus, have to settle for a good heuristic. For the heuristic we need the following notion

first used in [14]. Let K and M be sets of vectors in \mathbb{F}_2^n such that $K \subseteq M$. We say K is M -sum-free if no sum of two or more distinct vectors in K is equal to a vector in M . Observe that K is M -sum-free if and only if K is linearly independent and no sum of vectors in K is equal to a vector in $M \setminus K$.

The following lemma was proved implicitly in [14] and, thus, we provide a short proof of this result.

Lemma 10. *Let S be an irreducible system of MAX-LIN2-AA and let A be the matrix corresponding to S . Let M be the set of rows of A (viewed as vectors in \mathbb{F}_2^n) and let K be an M -sum-free set of k vectors. Let w_{\min} be the minimum weight of an equation in S . Then, in time in $(nm)^{O(1)}$, we can find an assignment to the variables of S that achieves excess of at least $w_{\min} \cdot k$.*

Proof. Let $\{e_{j_1}, \dots, e_{j_k}\}$ be the set of equations corresponding to the vectors in K . Run Algorithm \mathcal{H} , choosing at Step 1 an equation of S from $\{e_{j_1}, \dots, e_{j_k}\}$ each time, and let S' be the resulting system. Algorithm \mathcal{H} will run for k iterations of the while loop as no equation from $\{e_{j_1}, \dots, e_{j_k}\}$ will be deleted before it has been marked.

Indeed, suppose that this is not true. Then for some e_{j_i} and some other equation e in S , after applying Algorithm \mathcal{H} for at most $l - 1$ iterations e_{j_i} and e contain the same variables. Thus, there are vectors $v_j \in K$ and $v \in M$ and a pair of nonintersecting subsets K' and K'' of $K \setminus \{v, v_j\}$ such that $v_j + \sum_{u \in K'} u = v + \sum_{u \in K''} u$. Thus, $v = v_j + \sum_{u \in K' \cup K''} u$, a contradiction to the definition of K . □

5.2 Max- r -Lin2-AA

The following result was proved by Gutin et al. [30].

Theorem 7. *The problem MAX- r -LIN2-AA admits a kernel with at most $O(k^2)$ variables and equations.*

Proof. Let the system S be irreducible. Consider the excess

$$\varepsilon(x) = \sum_{I \in \mathcal{F}} w_I (-1)^{b_I} \prod_{i \in I} x_i. \tag{2}$$

Let us assign value -1 or 1 to each x_i with probability $1/2$ independently of the other variables. Then $X = \varepsilon(x)$ becomes a random variable. By Lemma 6, we have $\mathbb{E}(X^2) = \sum_{I \in \mathcal{F}} w_I^2$. Therefore, by Lemmas 3 and 4,

$$\mathbb{P}[X \geq \sqrt{m}/(2 \cdot 3^r)] \geq \mathbb{P} \left[X \geq \sqrt{\sum_{I \in \mathcal{F}} w_I^2}/(2 \cdot 3^r) \right] > 0.$$

Hence by Remark 1, if $\sqrt{m}/(2 \cdot 3^r) \geq 2k$, then the answer to MAX- r -LIN2-AA is YES. Otherwise, $m = O(k^2)$ and, by Rule 2, we have $n \leq m = O(k^2)$. □

The bound on the number of variables can be improved and it was done by Crowston et al. [14] and Kim and Williams [39]. The best known improvement is by Crowston, Fellows et al. [12]:

Theorem 8. *The problem MAX- r -LIN2-AA admits a kernel with at most $(2k - 1)r$ variables.*

This theorem can be easily proved using Formula (2), Lemma 10 and the following result by Crowston, Fellows et al. [12].

Lemma 11. *Let M be a set of vectors in \mathbb{F}_2^n such that M contains a basis of \mathbb{F}_2^n . Suppose that each vector of M contains at most r non-zero coordinates. If $k \geq 1$ is an integer and $n \geq r(k - 1) + 1$, then in time $|M|^{O(1)}$, we can find a subset K of M of k vectors such that K is M -sum-free.*

Both Theorem 8 and a slightly weaker analogous result of [39] imply the following:

Corollary 1. *There is an algorithm of runtime $2^{O(k)} + m^{O(1)}$ for MAX- r -LIN2-AA.*

Kim and Williams [39] proved that the last result is best possible, in a sense, if the Exponential Time Hypothesis holds.

Theorem 9. [39] *If MAX-3-LIN2-AA can be solved in $O(2^{\epsilon k} 2^{\epsilon m})$ time for every $\epsilon > 0$, then 3-SAT can be solved in $O(2^{\delta n})$ time for every $\delta > 0$, where n is the number of variables.*

5.3 Max- r -CSPs AA

Consider first a detailed formulation of MAX- r -CSP-AA. Let $V = \{v_1, \dots, v_n\}$ be a set of variables, each taking values -1 (TRUE) and 1 (FALSE). We are given a set Φ of Boolean functions, each involving at most r variables, and a collection \mathcal{F} of m Boolean functions, each $f \in \mathcal{F}$ being a member of Φ , each with a positive integral weight and each acting on some subset of V . We are to decide whether there is a truth assignment to the n variables such that the total weight of satisfied functions is at least $A + k$, where A is the average weight (over all truth assignments) of satisfied functions and k is the parameter.

Note that A is a tight lower bound for the problem, whenever the family Φ is closed under replacing each variable by its complement, since if we apply any Boolean function to all 2^r choices of literals whose underlying variables are any fixed set of r variables, then any truth assignment to the variables satisfies exactly the same number of these 2^r functions.

Note that if Φ consists of clauses, we get MAX- r -SAT-AA. In MAX- r -SAT-AA, $A = \sum_{j=1}^m w_j (1 - 2^{-r_j})$, where w_j and r_j are the weight and the number of variables of Clause j , respectively. Clearly, A is a tight lower bound for MAX- r -SAT.

Following [3], for a Boolean function f of weight $w(f)$ and on $r(f) \leq r$ Boolean variables $x_1, \dots, x_{r(f)}$, we introduce a polynomial $h_f(x)$, $x = (x_1, \dots, x_n)$ as follows. Let $S_f \subset \{-1, 1\}^{r(f)}$ denote the set of all satisfying assignments of f . Then

$$h_f(x) = w(f)2^{r-r(f)} \sum_{(v_1, \dots, v_{r(f)}) \in S_f} \prod_{j=1}^{r(f)} (1 + x_{i_j} v_j) - 1.$$

Let $h(x) = \sum_{f \in \mathcal{F}} h_f(x)$. It is easy to see (cf. [2]) that the value of $h(x)$ at some x^0 is precisely $2^r(U - A)$, where U is the total weight of the functions satisfied by the truth assignment x^0 . Thus, the answer to MAX- r -CSP-AA is YES if and only if there is a truth assignment x^0 such that $h(x^0) \geq k2^r$.

Algebraic simplification of $h(x)$ will lead us the following (Fourier expansion of $h(x)$, cf. [49]):

$$h(x) = \sum_{S \in \mathcal{F}} c_S \prod_{i \in S} x_i, \tag{3}$$

where $\mathcal{F} = \{\emptyset \neq S \subseteq [n] : c_S \neq 0, |S| \leq r\}$. Thus, $|\mathcal{F}| \leq n^r$. The sum $\sum_{S \in \mathcal{F}} c_S \prod_{i \in S} x_i$ can be viewed as the excess of an instance of MAX- r -LIN2-AA and, thus, we can reduce MAX- r -CSP-AA into MAX- r -LIN2-AA in polynomial time (since r is fixed, the algebraic simplification can be done in polynomial time and it does not matter whether the parameter of MAX- r -LIN2-AA is k or $k' = k2^r$). By Theorem 18, MAX- r -LIN2-AA has a kernel with $O(k^2)$ variables and equations. This kernel is a bikernel from MAX- r -CSP-AA to MAX- r -LIN2-AA. Thus, by Lemma 1, we obtain the following theorem of Alon et al. [2].

Theorem 10. MAX- r -CSP-AA admits a polynomial-size kernel.

Applying a reduction from MAX- r -LIN2-AA to MAX- r -SAT-AA in which each monomial in (3) is replaced by 2^{r-1} clauses, Alon et al. [2] obtained the following:

Theorem 11. MAX- r -SAT-AA admits a kernel with $O(k^2)$ clauses and variables.

Using also Theorem 8, it is easy to improve this theorem with respect to the number of variables in the kernel. This result was first obtained by Kim and Williams [39].

Theorem 12. MAX- r -SAT-AA admits a kernel with $O(k)$ variables.

6 MaxLin2-AA and MaxSat-AA

Recall that MAXLIN2-AA is the same problem as MAX- r -LIN2-AA, but the number of variables in an equation is not bounded. Thus, MAXLIN2-AA is a generalization of MAX- r -LIN2-AA. In this section we present a scheme of a recent proof by Crowston, et al. [12] that MAXLIN2-AA is fpt and has a kernel with polynomial number of variables. This result finally solved an open question

of Mahajan, Raman and Sikdar [45]. Still, we do not know whether MAXLIN2-AA has a kernel of polynomial size and we are able to give only partial results on the topic.

Theorem 13. [12] *The problem MAXLIN2-AA has a kernel with at most $O(k^2 \log k)$ variables.*

The proof of this theorem in [12] which we give later is based on Theorems 14 and 15.

Theorem 14. [14] *Let S be an irreducible system of MAXLIN2-AA and let $k \geq 2$. If $k \leq m \leq 2^{n/(k-1)} - 2$, then the maximum excess of S is at least k . Moreover, we can find an assignment with excess of at least k in time $m^{O(1)}$.*

This theorem can easily be proved using Lemma 10 and the following lemma.

Lemma 12. [14] *Let M be a set in \mathbb{F}_2^n such that M contains a basis of \mathbb{F}_2^n , the zero vector is in M and $|M| < 2^n$. If k is a positive integer and $k + 1 \leq |M| \leq 2^{n/k}$ then, in time $|M|^{O(1)}$, we can find an M -sum-free subset K of M with at least $k + 1$ vectors.*

Theorem 15. [12] *There exists an $n^{2k}(nm)^{O(1)}$ -time algorithm for MAXLIN2-AA that returns an assignment of excess of at least $2k$ if one exists, and returns NO otherwise.*

The proof of this theorem in [12] is based on constructing a special depth-bounded search tree.

Now we will present the proof of Theorem 14 from [12].

Proof of Theorem 14: Let \mathcal{L} be an instance of MAXLIN2-AA and let S be the system of \mathcal{L} with m equations and n variables. We may assume that S is irreducible. Let the parameter k be an arbitrary positive integer.

If $m < 2k$ then $n < 2k = O(k^2 \log k)$. If $2k \leq m \leq 2^{n/(2k-1)} - 2$ then, by Theorem 14 and Remark 1, the answer to \mathcal{L} is YES and the corresponding assignment can be found in polynomial time. If $m \geq n^{2k} - 1$ then, by Theorem 15, we can solve \mathcal{L} in polynomial time.

Finally we consider the case $2^{n/(2k-1)} - 2 \leq m \leq n^{2k} - 2$. Hence, $n^{2k} \geq 2^{n/(2k-1)}$. Therefore, $4k^2 \geq 2k + n/\log n \geq \sqrt{n}$ and $n \leq (2k)^4$. Hence, $n \leq 4k^2 \log n \leq 4k^2 \log(16k^4) = O(k^2 \log k)$.

Since S is irreducible, $m < 2^n$ and thus we have obtained the desired kernel. □

Now let us consider some cases where we can prove that MAXLIN2-AA has a polynomial-size kernel. Consider first the case when each equation in S has odd number of variables. Then we have the following theorem proved by Gutin et al. [30].

Theorem 16. *The special case of MAXLIN2-AA when each equation in S has odd number of variables, admits a kernel with at most $4k^2$ variables and equations.*

Proof. Let the system S be irreducible by Rule 1. Consider the excess $\epsilon(x) = \sum_{I \in \mathcal{F}} w_I (-1)^{b_I} \prod_{i \in I} x_i$. Let us assign value -1 or 1 to each x_i with probability $1/2$ independently of the other variables. Then $\epsilon(x)$ becomes a random variable. Since $\epsilon(-x) = -\epsilon(x)$, $\epsilon(x)$ is a symmetric random variable. Let $X = \epsilon(x)$. By Lemma 6, we have $\mathbb{E}(X^2) = \sum_{i \in I} w_I^2$. Therefore, by Lemma 2, $\mathbb{P}(X \geq \sqrt{m}) \geq \mathbb{P}(X \geq \sqrt{\sum_{j=1}^m w_j^2}) > 0$. Hence, if $\sqrt{m} \geq 2k$, the answer to MAXLIN2-AA is YES. Otherwise, $m < 4k^2$ and, after applying Rule 2, we have $n \leq m \leq 4k^2$. \square

In fact, Gutin et al. [30] proved the following more general result.

Theorem 17. *The following special case of MAXLIN2-AA admits a kernel with at most $4k^2$ variables and equations: there exists a subset U of variables such that each equation in $Ax = b$ has odd number of variables from U .*

Let us turn to results on MAXLIN2-AA that do not require any parity conditions. One such result is Theorem 7. Gutin et al. [30] also proved the following ‘dual’ theorem.

Theorem 18. *Let $\rho \geq 1$ be a fixed integer. Then MAXLIN2-AA restricted to instances where no variable appears in more than ρ equations, admits a kernel with $O(k^2)$ variables and equations.*

The proof is similar to that of Theorem 7, but Lemma 5 (in fact, its weaker version obtained in [30]) is used instead of Lemma 4.

Recall that MAXSAT-AA is the same problem as MAX- r -SAT-AA, but the number of variables in a clause is not bounded. Crowston et al. [15] proved that MAXSAT-AA is para-NP-complete and, thus, MAXSAT-AA is not fpt unless $P=NP$. This is in sharp contrast to MAXLIN2-AA. This result is a corollary of the following:

Theorem 19. [15] *MAX- $r(n)$ -SAT-AA is para-NP-complete for $r(n) = \lceil \log n \rceil$.*

The Exponential Time Hypothesis (ETH) claims that 3-SAT cannot be solved in time $2^{o(n)}$, where n is the number of variables (see, e.g., [22,48]). Using ETH, we can improve Theorem 19.

Theorem 20. [15] *Assuming ETH, MAX- $r(n)$ -SAT-AA is not fpt for any $r(n) \geq \log \log n + \phi(n)$, where $\phi(n)$ is any unbounded strictly increasing function of n .*

The following theorem shows that Theorem 20 provides a bound on $r(n)$ which is not far from optimal.

Theorem 21. [15] *MAX- $r(n)$ -SAT-AA is fpt for $r(n) \leq \log \log n - \log \log \log n - \phi(n)$, for any unbounded strictly increasing function $\phi(n)$.*

7 Ordering CSPs

In this section we will discuss recent results in the area of *Ordering Constraint Satisfaction Problems (Ordering CSPs)* parameterized above average. Ordering CSPs include several well-known problems such as BETWEENNESS, CIRCULAR ORDERING and ACYCLIC SUBDIGRAPH (which is equivalent to 2-LINEAR ORDERING). These three problems have applications in circuit design and computational biology [11,50], in qualitative spatial reasoning [35], and in economics [53], respectively.

Let us define Ordering CSPs of arity 3. The reader can easily generalize it to any arity $r \geq 2$ and we will do it below for LINEAR ORDERING of arity r . Let V be a set of n variables and let

$$\Pi \subseteq \mathcal{S}_3 = \{(123), (132), (213), (231), (312), (321)\}$$

be arbitrary. A *constraint set over V* is a multiset \mathcal{C} of *constraints*, which are permutations of three distinct elements of V . A bijection $\alpha : V \rightarrow [n]$ is called an *ordering* of V . For an ordering $\alpha : V \rightarrow [n]$, a constraint $(v_1, v_2, v_3) \in \mathcal{C}$ is Π -satisfied by α if there is a permutation $\pi \in \Pi$ such that $\alpha(v_{\pi(1)}) < \alpha(v_{\pi(2)}) < \alpha(v_{\pi(3)})$. Thus, given Π the problem Π -CSP, is the problem of deciding if there exists an ordering of V that Π -satisfies all the constraints. Every such problem is called an Ordering CSP of arity 3. We will consider the maximization version of these problems, denoted by MAX- Π -CSP, parameterized above the average number of constraints satisfied by a random ordering of V (which can be shown to be a tight bound).

Guttmann and Maucher [32] showed that there are in fact only 13 distinct Π -CSP's of arity 3 up to symmetry, of which 11 are nontrivial. They are listed in Table 1 together with their complexity. Note that if $\Pi = \{(123), (321)\}$ then we obtain the BETWEENNESS problem and if $\Pi = \{(123)\}$ then we obtain 3-LINEAR ORDERING.

Gutin et al. [27] proved that all 11 nontrivial MAX- Π -CSP problems are NP-hard (even though four of the Π -CSP are polynomial).

Now observe that given a variable set V and a constraint multiset \mathcal{C} over V , for a random ordering α of V , the probability of a constraint in \mathcal{C} being Π -satisfied by α equals $\frac{|\Pi|}{6}$. Hence, the expected number of satisfied constraints from \mathcal{C} is $\frac{|\Pi|}{6}|\mathcal{C}|$, and thus there is an ordering α of V satisfying at least $\frac{|\Pi|}{6}|\mathcal{C}|$ constraints (and this bound is tight). A derandomization argument leads to $\frac{|\Pi|}{6}$ -approximation algorithms for the problems MAX- Π_i -CSP [8]. No better constant factor approximation is possible assuming the Unique Games Conjecture [8].

We will study the parameterization of MAX- Π_i -CSP above tight lower bound:

Π -ABOVE AVERAGE (Π -AA)

Input: A finite set V of variables, a multiset \mathcal{C} of ordered triples of distinct variables from V and an integer $\kappa \geq 0$.

Parameter: κ .

Question: Is there an ordering α of V such that at least $\frac{|\Pi|}{6}|\mathcal{C}| + \kappa$ constraints of \mathcal{C} are Π -satisfied by α ?

Table 1. Ordering CSPs of arity 3 (after symmetry considerations)

$\Pi \subseteq \mathcal{S}_3$	Name	Complexity
$\Pi_0 = \{(123)\}$	LINEAR ORDERING-3	polynomial
$\Pi_1 = \{(123), (132)\}$		polynomial
$\Pi_2 = \{(123), (213), (231)\}$		polynomial
$\Pi_3 = \{(132), (231), (312), (321)\}$		polynomial
$\Pi_4 = \{(123), (231)\}$		NP-comp.
$\Pi_5 = \{(123), (321)\}$	BETWEENNESS	NP-comp.
$\Pi_6 = \{(123), (132), (231)\}$		NP-comp.
$\Pi_7 = \{(123), (231), (312)\}$	CIRCULAR ORDERING	NP-comp.
$\Pi_8 = \mathcal{S}_3 \setminus \{(123), (231)\}$		NP-comp.
$\Pi_9 = \mathcal{S}_3 \setminus \{(123), (321)\}$	NON-BETWEENNESS	NP-comp.
$\Pi_{10} = \mathcal{S}_3 \setminus \{(123)\}$		NP-comp.

In [27] it is shown that all 11 nontrivial Π -CSP-AA problems admit kernels with $O(\kappa^2)$ variables. This is shown by first reducing them to 3-LINEAR ORDERING-AA (or 2-LINEAR ORDERING-AA), and then finding a kernel for this problem, which is transformed back to the original problem. The first transformation is easy due to the following:

Proposition 1. [27] *Let Π be a subset of \mathcal{S}_3 such that $\Pi \notin \{\emptyset, \mathcal{S}_3\}$. There is a polynomial time transformation f from Π -AA to 3-LINEAR ORDERING-AA such that an instance (V, \mathcal{C}, k) of Π -AA is a YES-instance if and only if $(V, \mathcal{C}_0, k) = f(V, \mathcal{C}, k)$ is a YES-instance of 3-LINEAR ORDERING-AA.*

Proof. From an instance (V, \mathcal{C}, k) of Π -AA, construct an instance (V, \mathcal{C}_0, k) of 3-LINEAR ORDERING-AA as follows. For each triple $(v_1, v_2, v_3) \in \mathcal{C}$, add $|\Pi|$ triples $(v_{\pi(1)}, v_{\pi(2)}, v_{\pi(3)})$, $\pi \in \Pi$, to \mathcal{C}_0 .

Observe that a triple $(v_1, v_2, v_3) \in \mathcal{C}$ is Π -satisfied if and only if exactly one of the triples $(v_{\pi(1)}, v_{\pi(2)}, v_{\pi(3)})$, $\pi \in \Pi$, is satisfied by 3-LINEAR ORDERING. Thus, $\frac{|\Pi|}{6}|\mathcal{C}| + k$ constraints from \mathcal{C} are Π -satisfied if and only if the same number of constraints from \mathcal{C}_0 are satisfied by 3-LINEAR ORDERING. It remains to observe that $\frac{|\Pi|}{6}|\mathcal{C}| + k = \frac{1}{6}|\mathcal{C}_0| + k$ as $|\mathcal{C}_0| = |\Pi| \cdot |\mathcal{C}|$. \square

Recall that the maximization version of r -LINEAR ORDERING ($r \geq 2$) can be defined as follows. An instance of such a problem consists of a set of variables V and a multiset of constraints, which are ordered r -tuples of distinct variables of V (note that the same set of r variables may appear in several different constraints). The objective is to find an ordering α of V that maximizes the number of constraints whose order in α follows that of the constraint (we say that these constraints are satisfied). It is well-known that 2-LINEAR ORDERING is NP-hard (it follows immediately from the fact proved by Karp [37] that the feedback arc set problem is NP-hard). It is easy to extend this hardness result to

all r -LINEAR ORDERING problems (for each fixed $r \geq 2$). Note that in r -LINEAR ORDERING ABOVE AVERAGE (r -LINEAR ORDERING-AA), given a multiset \mathcal{C} of constraints over V we are to decide whether there is an ordering of V that satisfies at least $|\mathcal{C}|/r! + \kappa$ constraints.

(2,3)-LINEAR ORDERING is a mixture of 2-LINEAR ORDERING and 3-LINEAR ORDERING, where constraints can be of both arity 2 and 3.

We proceed by first considering 2-LINEAR ORDERING (Subsection 7.1), BETWEENNESS (Subsection 7.2), and 3-LINEAR ORDERING (Subsection 7.3) separately and proving the existence of a kernel with a quadratic number of variables and constraints for their parameterizations above average. We will conclude the section by briefly overviewing the result of Kim and Williams [39] that (2,3)-LINEAR ORDERING has a kernel with a linear number of variables (Subsection 7.4). By considering (2,3)-LINEAR ORDERING rather than just 3-LINEAR ORDERING separately, Kim and Williams managed to obtain a finite set of reduction rules which appear to be impossible to obtain for 3-LINEAR ORDERING only (see Subsection 7.3).

7.1 2-Linear Ordering

Let $D = (V, A)$ be a digraph on n vertices with no loops or parallel arcs in which every arc ij has a positive integral weight w_{ij} . Consider an ordering $\alpha : V \rightarrow [n]$ and the subdigraph $D_\alpha = (V, \{ij \in A : \alpha(i) < \alpha(j)\})$ of D . Note that D_α is acyclic. The problem of finding a subdigraph D_α of D of maximum weight is equivalent to 2-LINEAR ORDERING (where the arcs correspond to constraints and weights correspond to the number of occurrences of each constraint).

It is easy to see that, in the language of digraphs, 2-LINEAR ORDERING-AA can be formulated as follows.

2-LINEAR ORDERING ABOVE AVERAGE (2-LINEAR ORDERING-AA)

Instance: A digraph $D = (V, A)$, each arc ij has an integral positive weight w_{ij} , and a positive integer κ .

Parameter: The integer κ .

Question: Is there a subdigraph D_α of D of weight at least $W/2 + \kappa$, where $W = \sum_{ij \in A} w_{ij}$?

Mahajan, Raman, and Sikdar [45] asked whether 2-LINEAR ORDERING-AA is fpt for the special case when all arcs are of weight 1. Gutin et al. [30] solved the problem by obtaining a quadratic kernel for the problem. In fact, the problem can be solved using the following result of Alon [1]: there exists an ordering α such that D_α has weight at least $(\frac{1}{2} + \frac{1}{16|V|})W$. However, the proof in [1] uses a probabilistic approach for which a derandomization is not known yet and, thus, we cannot find the appropriate α deterministically. Moreover, the probabilistic approach in [1] is quite specialized. Thus, we briefly describe a solution from Gutin et al. [30] based on Strictly-Above-Below-Expectation Method (introduced in [30]).

Consider the following reduction rule:

Reduction Rule 3. Assume D has a directed 2-cycle iji ; if $w_{ij} = w_{ji}$ delete the cycle, if $w_{ij} > w_{ji}$ delete the arc ji and replace w_{ij} by $w_{ij} - w_{ji}$, and if $w_{ji} > w_{ij}$ delete the arc ij and replace w_{ji} by $w_{ji} - w_{ij}$.

It is easy to check that the answer to 2-Linear Ordering-AA for a digraph D is YES if and only if the answer to 2-Linear Ordering-AA is YES for a digraph obtained from D using the reduction rule as long as possible. A digraph is called an *oriented graph* if it has no directed 2-cycle. Note that applying Rule 3 as long as possible results in an oriented graph.

Consider a random ordering: $\alpha : V \rightarrow [n]$ and a random variable $X(\alpha) = \frac{1}{2} \sum_{ij \in A} x_{ij}(\alpha)$, where $x_{ij}(\alpha) = w_{ij}$ if $\alpha(i) < \alpha(j)$ and $x_{ij}(\alpha) = -w_{ij}$, otherwise. It is easy to see that $X(\alpha) = \sum \{w_{ij} : ij \in A, \alpha(i) < \alpha(j)\} - W/2$. Thus, the answer to 2-Linear Ordering-AA is YES if and only if there is an ordering $\alpha : V \rightarrow [n]$ such that $X(\alpha) \geq \kappa$. Since $\mathbb{E}(x_{ij}) = 0$, we have $\mathbb{E}(X) = 0$.

Let $W^{(2)} = \sum_{ij \in A} w_{ij}^2$. Gutin et al. [30] proved the following:

Lemma 13. *If D is an oriented graph, then $\mathbb{E}(X^2) \geq W^{(2)}/12$.*

Since $X(-\alpha) = -X(\alpha)$, where $-\alpha(i) = n + 1 - \alpha(i)$, X is a symmetric random variable and, thus, we use a proof similar to that of Theorem 16 (but applying Lemma 13 instead of Lemma 6) to show the following:

Theorem 22. *[30] 2-Linear Ordering-AA has a kernel with $O(\kappa^2)$ arcs.*

By deleting isolated vertices (if any), we can obtain a kernel with $O(\kappa^2)$ arcs and vertices. Kim and Williams [39] proved that 2-LINEAR ORDERING has a kernel with a linear number of variables.

7.2 Betweenness

Let $V = \{v_1, \dots, v_n\}$ be a set of variables and let \mathcal{C} be a multiset of m *betweenness* constraints of the form $(v_i, \{v_j, v_k\})$. For an ordering $\alpha : V \rightarrow [n]$, a constraint $(v_i, \{v_j, v_k\})$ is *satisfied* if either $\alpha(v_j) < \alpha(v_i) < \alpha(v_k)$ or $\alpha(v_k) < \alpha(v_i) < \alpha(v_j)$. In the BETWEENNESS problem, we are asked to find an ordering α satisfying the maximum number of constraints in \mathcal{C} . BETWEENNESS is NP-hard as even the problem of deciding whether all betweenness constraints in \mathcal{C} can be satisfied by an ordering α is NP-complete [50].

Let $\alpha : V \rightarrow [n]$ be a random ordering and observe that the probability of a constraint in \mathcal{C} to be satisfied is $1/3$. Thus, the expected number of satisfied constraints is $m/3$. A triple of betweenness constraints of the form $(v, \{u, w\}), (u, \{v, w\}), (w, \{v, u\})$ is called a *complete triple*. Instances of BETWEENNESS consisting of complete triples demonstrate that $m/3$ is a tight lower bound on the maximum number of constraints satisfied by an ordering α . Thus, the following parameterization is of interest:

BETWEENNESS ABOVE AVERAGE (BETWEENNESS-AA)

Instance: A multiset \mathcal{C} of m betweenness constraints over variables V and an integer $\kappa \geq 0$.

Parameter: The integer κ .

Question: Is there an ordering $\alpha : V \rightarrow [n]$ that satisfies at least $m/3 + \kappa$ constraints from \mathcal{C} ?

In order to simplify instances of BETWEENNESS-AA we introduce the following reduction rule.

Reduction Rule 4. *If \mathcal{C} has a complete triple, delete it from \mathcal{C} . Delete from V all variables that appear only in the deleted triple.*

Benny Chor’s question (see [48, p. 43]) to determine the parameterized complexity of BETWEENNESS-AA was solved by Gutin et al. [29] who proved that BETWEENNESS-AA admits a kernel with $O(\kappa^2)$ variables and constraints (in fact, [29] considers only the case when \mathcal{C} is a set, not a multiset, but the proof for the general case is the same [27]). Below we briefly describe the proof in [29].

Suppose we define a random variable $X(\alpha)$ just as we did for 2-LINEAR ORDERING. However such a variable is not symmetric and therefore we would need to use Lemma 6 on $X(\alpha)$. The problem is that α is a permutation and in Lemma 6 we are looking at polynomials, $f = f(x_1, x_2, \dots, x_n)$, over variables x_1, \dots, x_n each with domain $\{-1, 1\}$. In order to get around this problem the authors of [29] considered a different random variable $g(Z)$, which they defined as follows.

Let $Z = (z_1, z_2, \dots, z_{2n})$ be a set of $2n$ variables with domain $\{-1, 1\}$. These $2n$ variables correspond to n variables $z_1^*, z_2^*, \dots, z_n^*$ such that z_{2i-1} and z_{2i} form the binary representation of z_i^* . That is, z_i^* is 0, 1, 2 or 3 depending on the value of $(z_{2i-1}, z_{2i}) \in \{(-1, -1), (-1, 1), (1, -1), (1, 1)\}$. An ordering: $\alpha : V \rightarrow [n]$ complies with Z if for every $\alpha(i) < \alpha(j)$ we have $z_i^* \leq z_j^*$. We now define the value of $g(Z)$ as the average number of constraints satisfied over all orderings which comply with Z . Let $f(Z) = g(Z) - m/3$, and by Lemma 14 we can now use Lemma 6 on $f(Z)$ as it is a polynomial over variables whose domain is $\{-1, 1\}$. We consider variables z_i as independent uniformly distributed random variables and then $f(Z)$ is also a random variable. In [29] it is shown that the following holds if Reduction Rule 4 has been exhaustively applied.

Lemma 14. *The random variable $f(Z)$ can be expressed as a polynomial of degree 6. We have $\mathbb{E}[f(Z)] = 0$. Finally, if $f(Z) \geq \kappa$ for some $Z \in \{-1, 1\}^{2n}$ then the corresponding instance of BETWEENNESS-AA is a YES-instance.*

Lemma 15. [27] *For an irreducible (by Reduction Rule 4) instance we have $\mathbb{E}[f(Z)^2] \geq \frac{11}{768}m$.*

Theorem 23. [27] *BETWEENNESS-AA has a kernel of size $O(\kappa^2)$.*

Proof. Let (V, \mathcal{C}) be an instance of BETWEENNESS-AA. We can obtain an irreducible instance (V', \mathcal{C}') such that (V, \mathcal{C}) is a YES-instance if and only if (V', \mathcal{C}')

is a YES-instance in polynomial time. Let $m' = |\mathcal{C}'|$ and let $f(Z)$ be the random variable defined above. Then $f(Z)$ is expressible as a polynomial of degree 6 by Lemma 14; hence it follows from Lemma 4 that $\mathbb{E}[f(Z)^4] \leq 2^{36}\mathbb{E}[f(Z)^2]^2$. Consequently, $f(Z)$ satisfies the conditions of Lemma 3, from which we conclude that $\mathbb{P}\left(f(Z) > \frac{1}{4 \cdot 2^{18}} \sqrt{\frac{11}{768} m'}\right) > 0$, by Lemma 15. Therefore, by Lemma 14, if $\frac{1}{4 \cdot 2^{18}} \sqrt{\frac{11}{768} m'} \geq \kappa$ then (V', \mathcal{C}') is a YES-instance for BETWEENNESS-AA. Otherwise, we have $m' = O(\kappa^2)$. This concludes the proof of the theorem. \square

By deleting variables not appearing in any constraint, we obtain a kernel with $O(\kappa^2)$ constraints and variables.

7.3 3-Linear Ordering

In this subsection, we will give a short overview of the proof in [27] that 3-LINEAR ORDERING has a kernel with at most $O(\kappa^2)$ variables and constraints.

Unfortunately, approaches which we used for 2-LINEAR ORDERING-AA and BETWEENNESS-AA do not work for this problem. In fact, if we wanted to remove subsets of constraints where only the average number of constraints can be satisfied such that after these removals we are guaranteed to have more than the average number of constraints satisfied, then, in general case, an *infinite* number of reduction rules would be needed. The proof of this is quite long and therefore omitted from this survey, see [27] for more information.

However, we can reduce an instance of 3-LINEAR ORDERING-AA to instances of BETWEENNESS-AA and 2-LINEAR ORDERING-AA as follows. With an instance (V, \mathcal{C}) of 3-LINEAR ORDERING-AA, we associate an instance (V, \mathcal{B}) of BETWEENNESS-AA and two instances (V, A') and (V, A'') of 2-LINEAR ORDERING-AA such that if $C_p = (u, v, w) \in \mathcal{C}$, then add $B_p = (v, \{u, w\})$ to \mathcal{B} , $a'_p = (u, v)$ to A' , and $a''_p = (v, w)$ to A'' .

Let α be an ordering of V and let $\text{dev}(V, \mathcal{C}, \alpha)$ denote the number of constraints satisfied by α minus the average number of satisfied constraints in (V, \mathcal{C}) , where (V, \mathcal{C}) is an instance of 3-LINEAR ORDERING-AA, BETWEENNESS-AA or 2-LINEAR ORDERING-AA.

Lemma 16. [27] *Let (V, \mathcal{C}, κ) be an instance of 3-LINEAR ORDERING-AA and let α be an ordering of V . Then*

$$\text{dev}(V, \mathcal{C}, \alpha) = \frac{1}{2} [\text{dev}(V, A', \alpha) + \text{dev}(V, A'', \alpha) + \text{dev}(V, \mathcal{B}, \alpha)].$$

Therefore, we want to find an ordering satisfying as many constraints as possible from both of our new type of instances (note that we need to use the same ordering for all the problems).

Suppose we have a NO-instance of 3-LINEAR ORDERING-AA. As above, we replace it by three instances of BETWEENNESS-AA and 2-LINEAR ORDERING-AA. Now we apply the reduction rules for BETWEENNESS-AA and 2-LINEAR ORDERING-AA introduced above as well as the proof techniques described in

the previous sections in order to show that the total number of variables and constraints left in any of our instances is bounded by $O(\kappa^2)$. We then transform these reduced instances back into an instance of 3-LINEAR ORDERING-AA as follows. If $\{v, \{u, w\}\}$ is a BETWEENNESS constraint then we add the 3-LINEAR ORDERING-AA constraints (u, v, w) and (w, v, u) and if (u, v) is an 2-LINEAR ORDERING-AA constraint then we add the 3-LINEAR ORDERING-AA constraints (u, v, w) , (u, w, v) and (w, u, v) (for any $w \in V$). As a result, we obtain a kernel of 3-LINEAR ORDERING-AA with at most $O(\kappa^2)$ variables and constraints.

7.4 (2,3)-Linear Ordering-AA

In the previous subsection, we overviewed a result that 3-LINEAR ORDERING-AA has a kernel with at most $O(\kappa^2)$ variables and constraints. This result has been partially improved by Kim and Williams [39] who showed that 3-LINEAR ORDERING-AA has a kernel with at most $O(\kappa)$ variables. We will now outline their approach, where they considered (2,3)-LINEAR ORDERING-AA. That is, we allow constraints to contain 2 or 3 variables. Thus, we can apply the following reduction rules, where $w(e)$ denotes the weight of constraint e (i.e., the number of times e appears in the constraint multiset) and if $e = (u, v, w)$ is a constraint then we denote u by $e(1)$, v by $e(2)$ and w by $e(3)$, and $var(e)$ denotes the variables in e .

Redundancy Rule: Remove a variable v from V if it does not appear in any constraint. Remove a constraint e from C if its weight is zero.

Merging Rule: If e_1 and e_2 are identical, then replace them by a single constraint of weight $w(e_1) + w(e_2)$.

Cancellation Rule: If there are two constraints e_1, e_2 with $|e_1| = |e_2| = 2$ and $e_2 = (e_1(2), e_1(1))$, let $w_{\min} = \min\{w(e_1), w(e_2)\}$ and replace the weights by $w(e_1) = w(e_1) - w_{\min}$ and $w(e_2) = w(e_2) - w_{\min}$.

Edge Replacement Rule: If e_1, e_2, e_3 are three constraints in C with $var(e_1) = var(e_2) = var(e_3)$ and such that $e_2 = (e_1(2), e_1(1), e_1(3))$ and $e_3 = (e_1(1), e_1(3), e_1(2))$, then:

- replace the weight of a constraint by $w(e_i) = w(e_i) - w_{\min}$ for each $i = 1, 2, 3$, where $w_{\min} = \min\{w(e_1), w(e_2), w(e_3)\}$.
- add the binary ordering constraint $(e_1(1), e_1(3))$ of weight w_{\min} .

Cycle Replacement Rule: If e_1, e_2, e_3 are three constraints in C with $var(e_1) = var(e_2) = var(e_3)$ and such that $e_2 = (e_1(2), e_1(3), e_1(1))$ and $e_3 = (e_1(3), e_1(1), e_1(2))$, then:

- replace the weight of a constraint by $w(e_i) = w(e_i) - w_{\min}$ for each $i = 1, 2, 3$, where $w_{\min} = \min\{w(e_1), w(e_2), w(e_3)\}$.
- add the three binary ordering constraints $(e_1(1), e_1(2))$, $(e_1(2), e_1(3))$ and $(e_1(3), e_1(1))$, each of weight w_{\min} .

In [39] it is shown that these reduction rules produce equivalent instances. In [39] the following theorem is then proved.

Theorem 24. [39] *Let $I = (V, C, \kappa)$ be an irreducible (under the above reduction rules) instance of (2,3)-LINEAR ORDERING-AA. If I is a NO-instance (that is, less than $\rho W + \kappa$ constraints in I can be simultaneously satisfied, where ρW is the average weight of clauses satisfied by a random ordering), then the number of variables in I is $O(\kappa)$.*

In order to prove this theorem some above-mentioned techniques were used. Let $n = |V|$. As for BETWEENNESS-AA (see Subsection 7.2), Kim and Williams [39] introduced a random variable $f(y_1, \dots, y_{2n})$, which is a polynomial of degree 6 with $2n$ random uniformly distributed and independent variables y_i , each taking value 1 or -1 . The key property of $f(y_1, \dots, y_{2n})$ is that for every NO-instance I we have $f(y_1, \dots, y_{2n}) < \kappa$ for each $(y_1, \dots, y_{2n}) \in \{-1, 1\}^{2n}$. In Subsection 7.2, a similar inequality was used to bound the number of constraints in I using a probabilistic approach. Kim and Williams [39] use a different approach to bound the number of variables in I : they algebraically simplify $f(y_1, \dots, y_{2n})$ and obtain its Fourier expansion (see (3)). As in Subsection 5.3, the Fourier expansion can be viewed as the excess of the corresponding instance of MAX-6-LIN2-AA. Thus, to bound the number of variables in the Fourier expansion, we can use Theorem 8 (or, its weaker version obtained in [39]) which implies that the number is $O(\kappa)$.

However, there was a major obstacle that Kim and Williams [39] had to overcome. In general case, as a result of the algebraic simplification, the number of variables in the Fourier expansion may be significantly smaller than $2n$ and, thus, the bound on the number of variables in the Fourier expansion may not be used to bound n . To overcome the obstacle, Kim and Williams carefully analyzed the coefficients in the Fourier expansion and established that every variable of V is “represented” in the Fourier expansion. As a result, they concluded I can have only $O(\kappa)$ variables.

Acknowledgments. Research of Gutin was supported in part by the IST Programme of the European Community, under the PASCAL 2 Network of Excellence.

References

1. Alon, N.: Voting paradoxes and digraphs realizations. *Advances in Applied Math.* 29, 126–135 (2002)
2. Alon, N., Gutin, G., Kim, E.J., Szeider, S., Yeo, A.: Solving MAX- r -SAT above a tight lower bound. *Algorithmica* 61(3), 638–655 (2011)
3. Alon, N., Gutin, G., Krivelevich, M.: Algorithms with large domination ratio. *J. Algorithms* 50, 118–131 (2004)
4. Bang-Jensen, J., Gutin, G.: *Digraphs: Theory, Algorithms and Applications*, 2nd edn. Springer, London (2009)
5. Bodlaender, H.L., Downey, R.G., Fellows, M.R., Hermelin, D.: On problems without polynomial kernels. *J. Comput. Syst. Sci.* 75(8), 423–434 (2009)
6. Bodlaender, H.L., Thomassé, S., Yeo, A.: Kernel bounds for disjoint cycles and disjoint paths. *Theor. Comput. Sci.* 412(35), 4570–4578 (2011)
7. Bonami, A.: Étude des coefficients de Fourier des fonctions de $L^p(G)$. *Ann. Inst. Fourier* 20(2), 335–402 (1970)

8. Charikar, M., Guruswami, V., Manokaran, R.: Every permutation CSP of arity 3 is approximation resistant. In: Proc. Computational Complexity, pp. 62–73 (2009)
9. Chen, C., Kanj, I., Jia, W.: Vertex Cover: Further observations and further improvements. *J. Algorithms* 41, 280–301 (2001)
10. Chlebík, M., Clebiková, J.: Crown reductions for the Minimum Weighted Vertex Cover problem. *Discrete Appl. Math.* 156, 292–312 (2008)
11. Chor, B., Sudan, M.: A geometric approach to betweenness. *SIAM J. Discrete Math.* 11(4), 511–523 (1998)
12. Crowston, R., Fellows, M., Gutin, G., Jones, M., Rosamond, F., Thomassé, S., Yeo, A.: Simultaneously satisfying linear equations over \mathbb{F}_2 : MaxLin2 and Max-r-Lin2 parameterized above average. In: Chakraborty, S., Kumar, A. (eds.) FSTTCS 2011. *LIPICS*, vol. 13, pp. 229–240 (2011)
13. Crowston, R., Gutin, G., Jones, M.: Note on Max Lin-2 above average. *Inform. Proc. Lett.* 110, 451–454 (2010)
14. Crowston, R., Gutin, G., Jones, M., Kim, E.J., Ruzsa, I.Z.: Systems of Linear Equations over \mathbb{F}_2 and Problems Parameterized above Average. In: Kaplan, H. (ed.) SWAT 2010. *LNCS*, vol. 6139, pp. 164–175. Springer, Heidelberg (2010)
15. Crowston, R., Gutin, G., Jones, M., Raman, V., Saurabh, S.: Parameterized Complexity of MaxSat above Average. In: Fernández-Baca, D. (ed.) LATIN 2012. *LNCS*, vol. 7256, pp. 184–194. Springer, Heidelberg (2012)
16. Crowston, R., Gutin, G., Jones, M., Yeo, A.: A New Lower Bound on the Maximum Number of Satisfied Clauses in Max-SAT and Its Algorithmic Application. In: Raman, V., Saurabh, S. (eds.) IPEC 2010. *LNCS*, vol. 6478, pp. 84–94. Springer, Heidelberg (2010); *Algorithmica*, doi: 10.1007/s00453-011-9550-1
17. Cygan, M., Lokshtanov, D., Pilipczuk, M., Pilipczuk, M., Saurabh, S.: On the Hardness of Losing Width. In: Marx, D., Rossmanith, P. (eds.) IPEC 2011. *LNCS*, vol. 7112, pp. 159–168. Springer, Heidelberg (2012)
18. Downey, R.G., Fellows, M.R.: *Parameterized Complexity*. Springer (1999)
19. Fernau, H.: *Parameterized Algorithmics: A Graph-theoretic Approach*. Habilitation thesis, U. Tübingen (2005)
20. Fernau, H., Fomin, F.V., Lokshtanov, D., Raible, D., Saurabh, S., Villanger, Y.: Kernel(s) for problems with no kernel: On out-trees with many leaves. In: Proc. STACS 2009, pp. 421–432 (2009)
21. Fleischner, H., Kullmann, O., Szeider, S.: Polynomial-time recognition of minimal unsatisfiable formulas with fixed clause-variable difference. *Theoret. Comput. Sci.* 289(1), 503–516 (2002)
22. Flum, J., Grohe, M.: *Parameterized Complexity Theory*. Springer (2006)
23. Garey, M.R., Johnson, D.R.: *Computers and Intractability*. W.H. Freeman & Comp., New York (1979)
24. Gottlob, G., Szeider, S.: Fixed-parameter algorithms for artificial intelligence, constraint satisfaction, and database problems. *Comput. J.* 51(3), 303–325 (2008)
25. Guruswami, V., Håstad, J., Manokaran, R., Raghavendra, P., Charikar, M.: Beating the random ordering is hard: Every ordering CSP is approximation resistant. *SIAM J. Comput.* 40(3), 878–914 (2011)
26. Guruswami, V., Manokaran, R., Raghavendra, P.: Beating the random ordering is hard: Inapproximability of maximum acyclic subgraph. In: Proc. FOCS 2008, pp. 573–582 (2008)
27. Gutin, G., van Iersel, L., Mnich, M., Yeo, A.: Every ternary permutation constraint satisfaction problem parameterized above average has a kernel with a quadratic number of variables, *J. Comput. System Sci.* (in press), doi:10.1016/j.jcss.2011.01.004

28. Gutin, G., Jones, M., Yeo, A.: A New Bound for 3-Satisfiable Maxsat and Its Algorithmic Application. In: Owe, O., Steffen, M., Telle, J.A. (eds.) FCT 2011. LNCS, vol. 6914, pp. 138–147. Springer, Heidelberg (2011)
29. Gutin, G., Kim, E.J., Mnich, M., Yeo, A.: Betweenness parameterized above tight lower bound. *J. Comput. Syst. Sci.* 76, 872–878 (2010)
30. Gutin, G., Kim, E.J., Szeider, S., Yeo, A.: A probabilistic approach to problems parameterized above tight lower bound. *J. Comput. Syst. Sci.* 77, 422–429 (2011)
31. Gutin, G., Yeo, A.: Hypercontractive inequality for pseudo-Boolean functions of bounded Fourier width. *Discr. Appl. Math.* (to appear)
32. Guttmann, W., Maucher, M.: Variations on an ordering theme with constraints. In: Navarro, G., Bertossi, L., Kohayakawa, Y. (eds.) TCS 2006. IFIP, pp. 77–90. Springer, Boston (2006)
33. Håstad, J.: Some optimal inapproximability results. *J. ACM* 48, 798–859 (2001)
34. Håstad, J., Venkatesh, S.: On the advantage over a random assignment. *Random Structures & Algorithms* 25(2), 117–149 (2004)
35. Isli, A., Cohn, A.G.: A new approach to cyclic ordering of 2D orientations using ternary relation algebras. *Artif. Intelligence* 122(1-2), 137–187 (2000)
36. Jukna, S.: *Extremal Combinatorics With Applications in Computer Science*. Springer (2001)
37. Karp, R.M.: Reducibility among combinatorial problems. In: *Proc. Complexity of Computer Computations*. Plenum Press (1972)
38. Khot, S.: On the power of unique 2-prover 1-round games. In: *Proc. STOC 2002*, pp. 767–775 (2002)
39. Kim, E.J., Williams, R.: Improved Parameterized Algorithms for above Average Constraint Satisfaction. In: Marx, D., Rossmanith, P. (eds.) IPEC 2011. LNCS, vol. 7112, pp. 118–131. Springer, Heidelberg (2012)
40. Lampis, M.: A kernel of order $2k - c \log k$ for Vertex Cover. *Inf. Process. Lett.* 111(23-24), 1089–1091 (2011)
41. Lieberherr, K.J., Specker, E.: Complexity of partial satisfaction. *J. ACM* 28(2), 411–421 (1981)
42. Lieberherr, K.J., Specker, E.: Complexity of partial satisfaction, II. Tech. Report 293, Dept. of EECS. Princeton Univ. (1982)
43. Lokshtanov, D.: *New Methods in Parameterized Algorithms and Complexity*. PhD thesis, Bergen (2009)
44. Mahajan, M., Raman, V.: Parameterizing above guaranteed values: MaxSat and MaxCut. *J. Algorithms* 31(2), 335–354 (1999); Preliminary version in *Electr. Colloq. Comput. Complex (ECCC)*, TR-97-033 (1997)
45. Mahajan, M., Raman, V., Sikdar, S.: Parameterizing above or below guaranteed values. *J. Computer System Sciences* 75(2), 137–153 (2009); Mahajan, M., Raman, V., Sikdar, S.: Parameterizing MAX SNP Problems Above Guaranteed Values. In: Bodlaender, H.L., Langston, M.A. (eds.) IWPEC 2006. LNCS, vol. 4169, pp. 38–49. Springer, Heidelberg (2006)
46. Mishra, S., Raman, V., Saurabh, S., Sikdar, S., Subramanian, C.R.: The complexity of König subgraph problems and above-guarantee Vertex Cover. *Algorithmica* 61(4), 857–881 (2011)
47. Nemhauser, G.L., Trotter, L.E.: Vertex packings: structural properties and algorithms. *Math. Programming* 8(1), 232–248 (1975)
48. Niedermeier, R.: *Invitation to Fixed-Parameter Algorithms*. Oxford University Press (2006)

49. O'Donnell, R.: Some topics in analysis of Boolean functions. Technical report, ECCC Report TR08-055, Paper for an invited talk at STOC 2008 (2008), <http://www.eccc.uni-trier.de/eccc-reports/2008/TR08-055/>
50. Opatrný, J.: Total ordering problem. *SIAM J. Comput.* 8, 111–114 (1979)
51. Raman, V., Ramanujan, M.S., Saurabh, S.: Paths, Flowers and Vertex Cover. In: Demetrescu, C., Halldórsson, M.M. (eds.) *ESA 2011*. LNCS, vol. 6942, pp. 382–393. Springer, Heidelberg (2011)
52. Razgon, I., O'Sullivan, B.: Almost 2-SAT is fixed-parameter tractable. *J. Comput. Syst. Sci.* 75(8), 435–450 (2009)
53. Reinelt, G.: *The linear ordering problem: Algorithms and applications*. Heldermann Verlag (1985)
54. Soleimanfallah, A., Yeo, A.: A kernel of order $2k - c$ for Vertex Cover. *Discrete Math.* 311(10-11), 892–895 (2011)
55. Szeider, S.: Minimal unsatisfiable formulas with bounded clause-variable difference are fixed-parameter tractable. *J. Comput. Syst. Sci.* 69(4), 656–674 (2004)
56. Yannakakis, M.: On the approximation of maximum satisfiability. *J. Algorithms* 17, 475–502 (1994)

Backdoors to Satisfaction^{*}

Serge Gaspers and Stefan Szeider

Institute of Information Systems, Vienna University of Technology,
A-1040 Vienna, Austria

`gaspers@kr.tuwien.ac.at`, `stefan@szeider.net`

Dedicated to Mike Fellows on the occasion of his 60th birthday.

Abstract. A backdoor set is a set of variables of a propositional formula such that fixing the truth values of the variables in the backdoor set moves the formula into some polynomial-time decidable class. If we know a small backdoor set we can reduce the question of whether the given formula is satisfiable to the same question for one or several easy formulas that belong to the tractable class under consideration. In this survey we review parameterized complexity results for problems that arise in the context of backdoor sets, such as the problem of finding a backdoor set of size at most k , parameterized by k . We also discuss recent results on backdoor sets for problems that are beyond NP.

1 Introduction

Satisfiability (SAT) is the classical problem of determining whether a propositional formula in conjunctive normal form (CNF) has a satisfying truth assignment. The famous Cook-Levin Theorem [22,61], stating that SAT is NP-complete, placed satisfiability as the cornerstone of complexity theory. Despite its seemingly specialised nature, satisfiability has proved to be extremely useful in a wide range of different disciplines, both from the practical as well as from the theoretical point of view. Satisfiability provides a powerful and general formalism for solving various important problems including hardware and software verification and planning [8,64,99,56]. Satisfiability is the core of many reasoning problems in automated deduction; for instance, the package dependency management for the OpenSuSE Linux distribution and the autonomous controller for NASA's Deep Space One spacecraft are both based on satisfiability [6,100]. Over the last two decades, SAT-solvers have become amazingly successful in solving formulas with hundreds of thousands of variables that encode problems arising from various application areas, see, e.g., [50]. Theoretical performance guarantees, however, are far from explaining this empirically observed efficiency. In fact, there is an enormous gap between theory and practice. To illustrate it with numbers, take the exponential factor 1.308^n of the currently fastest known

^{*} Research supported by the European Research Council (ERC), project COMPLEX REASON 239962.

exact 3SAT algorithm [53]. Already for $n = 250$ variables this number exceeds by far the expected lifetime of the sun in nanoseconds.

Hidden Structure and Parameterized Complexity The discrepancy between theory and practice can be explained by the presence of a certain “hidden structure” in real-world problem instances. It is a widely accepted view that the structure of real-world problem instances makes the problems easy for heuristic solvers. However, classic worst-case analysis is not particularly well-suited to take this hidden structure into account. The classical model is one-dimensional, where only one aspect of the input (its size in bits, or the number of variables for a SAT formula) is taken into account, and it does not differentiate whether or not the instance is otherwise well-structured.

Parameterized Complexity, introduced by Mike Fellows together with Rod Downey offers a two-dimensional theoretical setting. The first dimension is the input size as usual, the second dimension (the parameter) allows to take structural properties of the problem instance into account. The result is a more fine-grained complexity analysis that has the potential of being more relevant to real-world computation while still admitting a rigorous theoretical treatment and firm algorithmic performance guarantees.

There are various ways of defining the “hidden structure” in a problem instance, yielding various ways to parameterize a problem.

Islands of Tractability. One way of coping with the high complexity of important problems within the framework of classical complexity is the identification of tractable sub-problems, i.e., of classes of instances for which the problem can be solved in polynomial time. Each class represents an “island of tractability” within an ocean of intractable problems. For the satisfiability problem, researchers have identified dozens of such islands – one could speak of an archipelago of tractability.

Usually it is quite unlikely that a real-world instance belongs to a known island of tractability, but it may be close to one. A very natural and humble way of parameterizing a problem is hence to take the distance to an island of tractability as a parameter. Guo *et al.* [52] called this approach “distance to triviality”. For SAT, the distance is most naturally measured in terms of the smallest number variables that need to be instantiated or deleted such that the instance gets moved to an island of tractability. Such sets of variables are called *backdoor sets* because once we know a small backdoor set we can solve the instance efficiently. Thus backdoor sets provide a “clever reasoning shortcut” through the search space and can be used as an indicator for the presence of a hidden structure in a problem instance. Backdoor sets were independently introduced by Crama *et al.* [27] and by Williams *et al.* [101], the latter authors coined the term “backdoor”.

The backdoor set approach to a problem consists of two steps: first a small backdoor set is computed (*backdoor detection*), second the backdoor set is used to solve the problem at hand (*backdoor evaluation*). It is hence natural to consider

an upper bound on the size of a smallest backdoor set as a parameter for both backdoor detection and backdoor evaluation.

2 Satisfiability

The *propositional satisfiability problem* (SAT) was the first problem shown to be NP-hard [22,61]. Despite its hardness, SAT solvers are increasingly leaving their mark as a general-purpose tool in areas as diverse as software and hardware verification, automatic test pattern generation, planning, scheduling, and even challenging problems from algebra [50].

A *literal* is a propositional variable x or a negated variable $\neg x$. We also use the notation $x = x^1$ and $\neg x = x^0$. A *clause* is a finite set literals that does not contain a complementary pair x and $\neg x$. A propositional formula in conjunctive normal form, or *CNF formula* for short, is a set of clauses. An r CNF formula is a CNF formula where each clause contains at most r literals. For a clause C we write $\text{var}(C) = \{x : x \in C \text{ or } \neg x \in C\}$, and for a CNF formula F we write $\text{var}(F) = \bigcup_{C \in F} \text{var}(C)$. An r -CNF formula is a CNF formula where each clause contains at most r literals. For a set S of literals we write $\overline{S} = \{x^{1-\epsilon} : x^\epsilon \in S\}$. We call a clause C *positive* if $C = \text{var}(C)$ and *negative* if $\overline{C} = \text{var}(C)$.

For a set X of propositional variables we denote by 2^X the set of all mappings $\tau : X \rightarrow \{0, 1\}$, the truth assignments on X . For $\tau \in 2^X$ we let $\text{true}(\tau) = \{x^{\tau(x)} : x \in X\}$ and $\text{false}(\tau) = \{x^{1-\tau(x)} : x \in X\}$ be the sets of literals set by τ to 1 and 0, respectively. Given a CNF formula F and a truth assignment $\tau \in 2^X$ we define $F[\tau] = \{C \in F : C \cap \text{true}(\tau) = \emptyset\}$. If $\tau \in 2^{\{x\}}$ and $\epsilon = \tau(x)$, we simply write $F[x = \epsilon]$ instead of $F[\tau]$.

A CNF formula F is *satisfiable* if there is some $\tau \in 2^{\text{var}(F)}$ with $F[\tau] = \emptyset$, otherwise F is *unsatisfiable*. Two CNF formulas are *equisatisfiable* if either both are satisfiable, or both are unsatisfiable. SAT is the NP-complete problem of deciding whether a given CNF formula is satisfiable [22,61].

Islands of Tractability and Backdoors. Backdoors are defined with respect to a fixed class \mathcal{C} of CNF formulas, the *base class* (or *target class*, or more figuratively, *island of tractability*). From a base class we require the following properties: (i) \mathcal{C} can be recognized in polynomial time, (ii) the satisfiability of formulas in \mathcal{C} can be decided in polynomial time, and (iii) \mathcal{C} is closed under isomorphisms (i.e., if two formulas differ only in the names of their variables, then either both or none belong to \mathcal{C}).

Several base classes considered in this survey also satisfy additional properties. Consider a class \mathcal{C} of CNF formulas. \mathcal{C} is *clause-induced* if it is closed under subsets, i.e., if $F \in \mathcal{C}$ implies $F' \in \mathcal{C}$ for each $F' \subseteq F$. \mathcal{C} is *clause-defined* if for each CNF formula F we have $F \in \mathcal{C}$ if and only if $\{C\} \in \mathcal{C}$ for all clauses $C \in F$. \mathcal{C} is closed under *variable-disjoint union* if for any two CNF formulas $F_1, F_2 \in \mathcal{C}$ with $\text{var}(F_1) \cap \text{var}(F_2) = \emptyset$, also $F_1 \cup F_2 \in \mathcal{C}$. \mathcal{C} is *self-reducible* if for any $F \in \mathcal{C}$ and any partial truth assignment τ , also $F[\tau] \in \mathcal{C}$.

A *strong \mathcal{C} -backdoor set* of a CNF formula F is a set B of variables such that $F[\tau] \in \mathcal{C}$ for each $\tau \in 2^B$. A *weak \mathcal{C} -backdoor set* of F is a set B of

variables such that $F[\tau]$ is satisfiable and $F[\tau] \in \mathcal{C}$ holds for some $\tau \in 2^B$. A *deletion \mathcal{C} -backdoor set* of F is a set B of variables such that $F - B \in \mathcal{C}$, where $F - B = \{C \setminus \{x^0, x^1 : x \in B\} : C \in F\}$.

If we know a strong \mathcal{C} -backdoor set of F of size k , we can reduce the satisfiability of F to the satisfiability of 2^k formulas in \mathcal{C} . Thus SAT becomes fixed-parameter tractable in k . If we know a weak \mathcal{C} -backdoor set of F , then F is clearly satisfiable, and we can verify it by trying for each $\tau \in 2^X$ whether $F[\tau]$ is in \mathcal{C} and satisfiable. If \mathcal{C} is clause-induced, any deletion \mathcal{C} -backdoor set of F is a strong \mathcal{C} -backdoor set of F . For several base classes, deletion backdoor sets are of interest because they are easier to detect than strong backdoor sets. The challenging problem is to find a strong, weak, or deletion \mathcal{C} -backdoor set of size at most k if it exists. For each class \mathcal{C} of CNF formulas we consider the following decision problems.

STRONG \mathcal{C} -BACKDOOR SET DETECTION

Instance: A CNF formula F and an integer $k \geq 0$.

Parameter: The integer k .

Question: Does F have a strong \mathcal{C} -backdoor set of size at most k ?

The problems WEAK \mathcal{C} -BACKDOOR SET DETECTION and DELETION \mathcal{C} -BACKDOOR SET DETECTION are defined similarly.

In fact, for the backdoor approach we actually need the functional variants of these problems, where if a backdoor set of size at most k exists, such a set is computed. However, for all cases considered in this survey, where backdoor detection is fixed-parameter tractable, the respective algorithms also compute a backdoor set.

We also consider these problems for formulas with bounded clause lengths. All such results are stated for 3CNF formulas, but hold, more generally, for r CNF formulas, where $r \geq 3$ is a fixed integer.

3 Base Classes

In this section we define the base classes for the SAT problem that we will consider in this survey.

3.1 Schaefer's Base Classes

In his seminal paper, Schaefer [93] classified the complexity of generalized satisfiability problems in terms of the relations that are allowed to appear in constraints. For CNF satisfiability, this yields the following five base classes¹.

1. *Horn formulas:* CNF formulas where each clause contains at most one positive literal.
2. *Anti-Horn formulas:* CNF formulas where each clause contains at most one negative literal.

¹ Affine Boolean formulas considered by Schaefer do not correspond naturally to a class of CNF formulas, hence we do not consider them here.

3. *2CNF formulas*: CNF formulas where each clause contains at most two literals.
4. *0-valid formulas*: CNF formulas where each clause contains at least one negative literal.
5. *1-valid formulas*: CNF formulas where each clause contains at least one positive literal.

We denote the respective classes of CNF formulas by HORN , HORN^- , 2CNF , 0-VAL , and 1-VAL , and we write $\text{Schaefer} = \{\text{HORN}, \text{HORN}^-, 2\text{CNF}, 0\text{-VAL}, 1\text{-VAL}\}$. We note that all these classes are clause-defined, and by Schaefer's Theorem, these are the only maximal clause-defined base classes. We also note that 0-VAL and 1-VAL are the only two base classes considered in this survey that are not self-reducible.

3.2 Base Classes Based on Subsolvers

State-of-the-art SAT-solvers are based on variants of the so-called Davis-Logemann-Loveland (DPLL) procedure [29,30] (see also [23]). The DPLL procedure searches systematically for a satisfying assignment, applying first *unit propagation* and *pure literal elimination* as often as possible. Then, DPLL branches on the truth value of a variable, and recurses. The algorithm stops if either there are no clauses left (the original formula is satisfiable) or all branches of the search lead to an empty clause (the original formula is unsatisfiable). Unit propagation takes as input a CNF formula F that contains a "unit clause" $\{x^\epsilon\}$ and outputs $F[x = \epsilon]$. Pure literal elimination takes as input a CNF formula F that has a "pure literal" x^ϵ , where $x \in \text{var}(F)$ and $x^{1-\epsilon} \notin \bigcup_{C \in F} C$, and outputs $F[x = \epsilon]$. In both cases F and $F[x = \epsilon]$ are equisatisfiable. If we omit the branching, we get an incomplete algorithm which decides satisfiability for a subclass of CNF formulas. Whenever the algorithm reaches the branching step, it halts and outputs "give up". This incomplete algorithm is an example of a "subsolver" as considered by Williams *et al.* [101]. The DPLL procedure gives rise to three non-trivial subsolvers: UP + PL (unit propagation and pure literal elimination are available), UP (only unit propagation is available), PL (only pure literal elimination is available). We associate each subsolver with the class of CNF formulas for which it determines the satisfiability (this is well-defined, since unit propagation and pure literal elimination are confluent operations). Since the subsolvers clearly run in polynomial time, UP + PL, UP, and PL form base classes. We write $\text{Subsolver} = \{\text{UP} + \text{PL}, \text{UP}, \text{PL}\}$.

3.3 Miscellaneous Base Classes

Renamable Horn Let X be a set of variables and F a CNF formula. We let $r_X(F)$ denote the CNF formula obtained from F by replacing for every variable $x \in X$, all occurrences of x^ϵ in F with $x^{1-\epsilon}$, for $\epsilon \in \{0, 1\}$. We call $r_X(F)$ a *renaming* of F . Clearly F and $r_X(F)$ are equisatisfiable. A CNF formula is called *renamable Horn* if it has a renaming which is Horn, and we denote the

Table 1. The parameterized complexity of WEAK, STRONG, and DELETION \mathcal{C} -BACKDOOR SET DETECTION for various base classes \mathcal{C}

Base Class	WEAK	STRONG	DELETION
$\mathcal{C} \in \text{Schaefer}$	W[2]-h ^[70] (FPT)	FPT ^[70]	FPT ^[70]
$\mathcal{C} \in \text{Subsolver}$	W[P]-c ^[96]	W[P]-c ^[96]	n/a
FOREST	W[2]-h ^[46] (FPT ^[46])	? [†] (?)	FPT
RHORN	W[2]-h	W[2]-h (?)	FPT ^[82]
CLU	W[2]-h ^[71] (FPT)	W[2]-h ^[71] (FPT ^[71])	FPT ^[71]

^() It is indicated in parentheses if the complexity of the problem for 3CNF formulas is different from general CNF or unknown.

[?] It is open whether the problem is fixed-parameter tractable.

[†] Theorem 5 gives an fpt approximation for this problem.

^{n/a} Deletion backdoor sets are undefined for base classes that are not clause-induced.

class of renamable Horn formulas as RHORN. It is easy to see that HORN is a strict subset of RHORN. One can find in polynomial time a Horn renaming of a given CNF formula, if it exists [62]. Hence RHORN is a further base class. In contrast to HORN, RHORN is not clause-defined.

Forests. Many NP-hard problems can be solved in polynomial time for problem instances that are in a certain sense acyclic. The satisfiability problem is no exception. There are various ways of defining a CNF formula to be acyclic. Here we consider acyclicity based on (undirected) incidence graphs: the *incidence graph* of a CNF formula F is the bipartite graph whose vertices are the variables and the clauses of F ; a variable x and a clause C are joined by an edge if and only if $x \in \text{var}(C)$. Let FOREST denote the class of CNF formulas whose undirected incidence graphs are forests. It is well known that FOREST forms islands of tractability: the satisfiability of CNF formulas whose incidence graphs have bounded treewidth can be decided in linear time [43,91]. FOREST is the special case of formulas with treewidth at most 1.

Clusters. A CNF formula F is called a *hitting* if any two distinct clauses clash. Two clauses $C, C' \in F$ *clash* if they contain a complimentary pair of literals, i.e., $C \cap \overline{C'} \neq \emptyset$. A CNF formula is called a *clustering formula* if it is a variable disjoint union of hitting formulas. We denote by CLU the class of clustering formulas. Clustering formulas not only allow polynomial-time SAT decision, one can even count the number of satisfying truth assignments in polynomial time. This is due to the fact that each truth assignment invalidates at most one clause of a hitting formula [54,71].

4 Detecting Weak Backdoor Sets

It turns out that for all base classes \mathcal{C} considered in this survey, WEAK \mathcal{C} -BACKDOOR SET DETECTION is W[2]-hard. In several cases, restricting the input

formula to 3CNF helps, and makes WEAK \mathcal{C} -BACKDOOR SET DETECTION fixed-parameter tractable.

In the proof of the following proposition we use a general approach that entails previously published proofs (such as in [46,70,71]) as special cases.

Proposition 1. WEAK \mathcal{C} -BACKDOOR SET DETECTION is $W[2]$ -hard for all base classes $\mathcal{C} \in \text{Schaefer} \cup \{\text{RHORN}, \text{FOREST}, \text{CLU}\}$.

Proof. We show $W[2]$ -hardness for $\mathcal{C} \in \{2\text{CNF}, \text{HORN}, 0\text{-VAL}, \text{RHORN}, \text{FOREST}, \text{CLU}\}$. The hardness proofs for the remaining two classes 1-VAL and HORN^- are symmetric to the proofs for 0-VAL and HORN, respectively.

Let G be a CNF formula with a set $X \subseteq \text{var}(G)$ of its variables marked as *external*, all other variables of G are called *internal*. We call G an *or-gadget* for a base class \mathcal{C} if G has the following properties:

1. $G \notin \mathcal{C}$.
2. $G \in 1\text{-VAL}$.
3. $G[x = 1] \in \mathcal{C}$ holds for all $x \in X$.
4. For each clause $C \in G$ either $X \subseteq C$ or $\text{var}(C) \cap X = \emptyset$.
5. $\text{var}(G) \setminus X \neq \emptyset$.
6. G can be constructed in time polynomial in $|X|$.

First, we show the following meta-result, and then we define or-gadgets for the different base-classes.

Claim 1: If \mathcal{C} is clause-induced, closed under variable-disjoint union, and has an or-gadget for any number ≥ 1 of external variables, then WEAK \mathcal{C} -BACKDOOR SET DETECTION is $W[2]$ -hard.

We prove the claim by giving a parameterized reduction from the $W[2]$ -complete problem HITTING SET (HS) [33]. Let (\mathcal{S}, k) , $\mathcal{S} = \{S_1, \dots, S_m\}$, be an instance of HS. Let $I = \{1, \dots, m\} \times \{1, \dots, k + 1\}$. For each S_i we construct $k + 1$ or-gadgets G_i^1, \dots, G_i^{k+1} whose external variables are exactly the elements of S_i , and whose internal variables do not appear in any of the other gadgets $G_{i'}^{j'}$ for $(i', j') \in I \setminus \{(i, j)\}$. Let $F = \bigcup_{(i,j) \in I} G_i^j$. From Property 6 it follows that F can be constructed from \mathcal{S} in polynomial time. We show that \mathcal{S} has a hitting set of size k if and only if F has a weak \mathcal{C} -backdoor set of size k .

Assume $B \subseteq \bigcup_{i=1}^m S_i$ is a hitting set of \mathcal{S} of size k . Let $\tau \in 2^B$ the truth assignment that sets all variables from B to 1. By Properties 2 and 3, $G_i^j[\tau]$ is satisfiable and belongs to \mathcal{C} for each $(i, j) \in I$. By Property 4, $\text{var}(G_i^j[\tau]) \cap \text{var}(G_{i'}^{j'}[\tau]) = \emptyset$ for any two distinct pairs $(i, j), (i', j') \in I$. Consequently $F[\tau]$ is satisfiable, and since \mathcal{C} is closed under variable-disjoint union, $F[\tau]$ belongs to \mathcal{C} . Thus B is a weak \mathcal{C} -backdoor set of F of size k .

Conversely, assume that $B \subseteq \text{var}(F)$ is a weak \mathcal{C} -backdoor set of F of size k . Hence, there exists a truth assignment $\tau \in 2^B$ such that $F[\tau]$ is satisfiable and belongs to \mathcal{C} . Clearly for each $(i, j) \in I$, $G_i^j[\tau]$ is satisfiable (since $G_i^j[\tau] \subseteq F$), and $G_i^j[\tau] \in \mathcal{C}$ (since \mathcal{C} is clause-induced). However, since $G_i^j \notin \mathcal{C}$ by Property 1, $B \cap \text{var}(G_i^j) \neq \emptyset$ for each $(i, j) \in I$. Let $1 \leq i \leq m$. By construction, F contains

$k + 1$ copies G_i^1, \dots, G_i^{k+1} of the same gadget. From Property 5 it follows that all the $k + 1$ copies are different. Since $|B| \leq k$, there must be some $x_i \in B$ such that there are $1 \leq j' < j'' \leq k + 1$ with $x_i \in \text{var}(G_i^{j'}) \cap \text{var}(G_i^{j''})$. It follows that x_i is an external variable of $G_i^{j'}$, hence $x_i \in B \cap S_i$. Consequently, B is a hitting set of \mathcal{S} .

Hence we have indeed a parameterized reduction from HS to WEAK \mathcal{C} -BACKDOOR SET DETECTION, and Claim 1 is shown true. We define for each class $\mathcal{C} \in \{2\text{CNF}, \text{HORN}, 0\text{-VAL}, \text{RHORN}, \text{FOREST}, \text{CLU}\}$ an or-gadget $F(\mathcal{C})$ where $X = \{x_1, \dots, x_s\}$ is the set of external variables; internal variables are denoted z_i .

- $G(2\text{CNF}) = \{X \cup \{z_1, z_2\}\}$.
- $G(\text{HORN}) = G(0\text{-VAL}) = \{X \cup \{z_1\}\}$.
- $G(\text{RHORN}) = \{X \cup \{\neg z_1, \neg z_2\}, \{z_1, \neg z_2\}, \{\neg z_1, z_2\}, \{z_1, z_2\}\}$.
- $G(\text{FOREST}) = \{X \cup \{\neg z_1, \neg z_2\}, \{z_1, z_2\}\}$.
- $G(\text{CLU}) = \{X \cup \{z_1\}, \{z_1\}\}$.

Since the considered classes \mathcal{C} are clearly clause-induced and closed under variable-disjoint union, the proposition now follows from Claim 1. □

For base classes based on subsolvers, weak backdoor set detection is even W[P]-hard. This is not surprising, since the subsolvers allow a propagation through the formula which is similar to the propagation in problems like MINIMUM AXIOM SET or DEGREE 3 SUBGRAPH ANNIHILATOR [33]. The proof of the following theorem is based on a reduction from the W[P]-complete problem CYCLIC MONOTONE CIRCUIT ACTIVATION.

Theorem 1 ([96]). WEAK \mathcal{C} -BACKDOOR SET DETECTION is W[P]-complete for all base classes $\mathcal{C} \in \text{Subsolver}$. This even holds if the input formula is in 3CNF.

In summary, we conclude that WEAK \mathcal{C} -BACKDOOR SET DETECTION is at least W[2]-hard for all considered base classes. If we restrict our scope to 3CNF formulas, we obtain mixed results.

Proposition 2. For every clause-defined class \mathcal{C} , WEAK \mathcal{C} -BACKDOOR SET DETECTION is fixed-parameter tractable for input formulas in 3CNF.

Proof. The result follows by a standard bounded search tree argument, sketched as follows. Assume we are given a CNF formula $F \notin \mathcal{C}$ and an integer k . We want to decide whether F has a weak \mathcal{C} -backdoor set of size $\leq k$. Since \mathcal{C} is clause-defined, F contains a clause C such that $\{C\} \notin \mathcal{C}$. Hence some variable of $\text{var}(C)$ must belong to any weak \mathcal{C} -backdoor set of F . There are at most 3 such variables, each of which can be set to true or to false. Hence we branch in at most 6 cases. By iterating this case distinction we build a search tree T , where each node t of T corresponds to a partial truth assignment τ_t . We can stop building the tree at nodes of depth k and at nodes t where $F[\tau_t] \in \mathcal{C}$. It is now easy to see that F has a weak \mathcal{C} -backdoor set of size at most k if and only

if T has a leaf t such that $F[\tau_t] \in \mathcal{C}$ and $F[\tau_t]$ is satisfiable. For each leaf we can check in polynomial time whether these properties hold. \square

In particular, WEAK \mathcal{C} -BACKDOOR SET DETECTION is fixed-parameter tractable for $\mathcal{C} \in \text{Schaefer}$ if the input formula is in 3CNF.

The proof of Proposition 2 can be extended to the class CLU of clustering formulas. Nishimura et al. [71] have shown that a CNF formula is a clustering formula if and only if it does not contain (i) two clauses C_1, C_2 that overlap ($C_1 \cap C_2 \neq \emptyset$) but do not clash ($C_1 \cap \overline{C_2} = \emptyset$), or (ii) three clauses D_1, D_2, D_3 where D_1 and D_2 clash, D_2 and D_3 clash, but D_1 and D_3 do not clash. $\{C_1, C_2\}$ is called an overlap obstruction, $\{D_1, D_2, D_3\}$ is called a clash obstruction. Each weak CLU-backdoor set of a CNF formula F must contain at least one variable from each overlap and each clash obstruction. However, if F is a 3CNF formula, the number of variables of an overlap obstruction is at most 5, and the number of variables of a clash obstruction is at most 7. Hence we can find a weak CLU-backdoor set of size at most k with a bounded search tree, which gives the following result.

Proposition 3. WEAK CLU-BACKDOOR SET DETECTION is fixed-parameter tractable for 3CNF formulas.

Proposition 4. WEAK RHORN-BACKDOOR SET DETECTION is W[2]-hard, even for 3CNF formulas.

Proof. Similarly to the proof of Proposition 1 we reduce from HS. As gadgets we use formulas of the form $G = \{\{z_1, \neg x_1, \neg z_2\}, \{z_2, \neg x_2, \neg z_3\}, \dots, \{z_s, \neg x_s, \neg z_{s+1}\}, \{\neg z_1, z_{s+1}\}, \{\neg z_1, \neg z_{s+1}\}, \{z_1, z_{s+1}\}\}$, where x_1, \dots, x_s are external variables and z_1, \dots, z_{s+1} are internal variables. G can be considered as being obtained from the complete formula $\{\{z_1^\epsilon, z_{s+1}^\delta\} : \epsilon, \delta \in \{0, 1\}\}$ by “subdividing” the clause $\{z_1, \neg z_{s+1}\}$. $G \notin \text{RHORN}$ but $G[x_i = 0] \in \text{RHORN}$. In fact, $r_X(G[x_i = 0]) \in \text{HORN}$ for $X = \{z_{i+1}, \dots, z_{s+1}\}$, hence no external variable needs to be renamed. Moreover, we can satisfy $G[x_i = 0]$ by setting all external variables and z_1 to 0, and by setting z_{s+1} to 1.

Let (\mathcal{S}, k) , $\mathcal{S} = \{S_1, \dots, S_m\}$, be an instance of HS. For each S_i we construct $k + 1$ gadgets G_i^1, \dots, G_i^{k+1} , each having S_i as the set of its external variables, and the internal variables are new variables only used inside a gadget. We let F to be the union of all such gadgets G_i^j for $1 \leq i \leq m$ and $1 \leq j \leq k + 1$.

Similar to the proof of Proposition 1 we can easily show that \mathcal{S} has a hitting set of size k if and only if F has a weak RHORN-backdoor set of size k . The proposition follows. \square

According to Propositions 2 and 3, WEAK \mathcal{C} -BACKDOOR SET DETECTION is fixed-parameter tractable for certain base classes \mathcal{C} and input formulas in 3CNF. For the classes \mathcal{C} covered by Propositions 2 and 3 it holds that for every 3CNF formula $F \notin \mathcal{C}$ we can find a set of variables of bounded size, an “obstruction”, from which at least one variable must be in any weak \mathcal{C} -backdoor set of F . Hence a weak \mathcal{C} backdoor set of size at most k can be found by means of a bounded search tree algorithm. The next result shows that fixed-parameter tractability

also prevails for the base class FOREST. However, the algorithm is considerably more complicated, as in this case we do not have obstructions of bounded size.

Theorem 2 ([46]). WEAK FOREST-BACKDOOR SET DETECTION is fixed-parameter tractable for 3CNF formulas.

Proof (Sketch). We sketch the fpt algorithm from [46] deciding whether a 3CNF formula has a weak FOREST-backdoor set of size k . We refer to [46] for the full details and the correctness proof. Let G denote the incidence graph of F . The first step of the algorithm runs an fpt algorithm (with parameter k') by Bodlaender [9] that either finds $k' = 2k + 1$ vertex-disjoint cycles in G or a feedback vertex set of G of size at most $12k'^2 - 27k' + 15$.

In case a feedback vertex set X is returned, a tree decomposition of $G \setminus X$ of width 1 is computed and X is added to each bag of this tree decomposition. As the WEAK FOREST-BACKDOOR SET DETECTION problem can be defined in Monadic Second Order Logic, a meta-theorem by Courcelle [26] can use this tree decomposition to conclude.

In case Bodlaender's algorithm returns k' vertex-disjoint cycles, the algorithm finds a set S^* of $O(4^k k^6)$ variables such that any weak FOREST-backdoor set of size k contains at least one variable from S^* . In this case, the algorithm recurses by considering all possibilities of assigning a value to a variable from S^* .

Let $C_1, \dots, C_{k'}$ denote the variable-disjoint cycles returned by Bodlaender's algorithm. Consider a variable $x \in \text{var}(F)$ and a cycle C . We say that x kills C internally if $x \in C$. We say that x kills C externally if $x \notin C$ and C contains a clause $u \in F$ such that $x \in \text{var}(u)$.

As our k' cycles are all vertex-disjoint, at most k cycles may be killed internally. The algorithm goes through all choices of k cycles among $C_1, \dots, C_{k'}$ that may be killed internally. All other cycles, say C_1, \dots, C_{k+1} , are not killed internally and need to be killed externally. The algorithm now computes a set $S \subseteq \text{var}(F)$ of size $O(k^6)$ such that any weak FOREST-backdoor set of size k , which is a subset of $\text{var}(F) \setminus \bigcup_{i=1}^{k+1} \text{var}(C_i)$, contains at least one variable from S . The union of all such S , taken over all choices of cycles to be killed internally, forms then the set S^* that was to be computed.

For each cycle from C_1, \dots, C_{k+1} , compute its set of external killers in $\text{var}(F) \setminus \bigcup_{i=1}^{k+1} \text{var}(C_i)$. Only these external killers are considered from now on. If one such cycle has no such external killer, then there is no solution with the current specifications and the algorithm backtracks. For each $i, 1 \leq i \leq k + 1$, let x_i denote an external killer of C_i with a maximum number of neighbors in C_i . The algorithm executes the first applicable from the following rules.

Multi-Killer Unsupported. If there is an index $i, 1 \leq i \leq k + 1$ such that x_i has $\ell \geq 4k$ neighbors in C_i and at most $4k^2 + k$ external killers of C_i have at least $\ell/(2k)$ neighbors in C_i , then include all these external killers in S .

Multi-Killer Supported. If there is an index $i, 1 \leq i \leq k + 1$ such that x_i has $\ell \geq 4k$ neighbors in C_i and more than $4k^2 + k$ external killers of C_i have at least $\ell/(2k)$ neighbors in C_i , then set $S = \{x_i\}$.

Large Overlap. If there are two cycles $C_i, C_j, 1 \leq i \neq j \leq k + 1$, with at least $16k^4 + k$ common external killers, then set $S = \emptyset$.

Small Overlap. Otherwise, include in S all vertices that are common external killers of at least two cycles from C_1, \dots, C_{k+1} .

The algorithm recursively checks for each $s \in S^*$ whether the formulas $F[s = 0]$ and $F[s = 1]$ have a weak FOREST-backdoor set of size $k - 1$ and returns YES if any such recursive call was successful and NO otherwise. \square

5 Detecting Strong Backdoor Sets

Proposition 5 ([70]). STRONG \mathcal{C} -BACKDOOR SET DETECTION is fixed-parameter tractable for every base class $\mathcal{C} \in \text{Schaefer}$. For $\mathcal{C} \in \{0\text{-VAL}, 1\text{-VAL}\}$, the problem is even solvable in polynomial time.

Proof. Consider a CNF formula F . Strong HORN-backdoor sets of F are exactly the vertex covers of the positive primal graph of F , whose vertex set is $\text{var}(F)$, two variables are joined by an edge if they appear together positively in a clause. Strong HORN⁻-backdoor sets can be characterized symmetrically. Strong 2CNF-backdoor sets of F are exactly the hitting sets of the hypergraph whose vertex set is $\text{var}(F)$ and whose hyperedges are all the subsets $e \subseteq \text{var}(F)$ of size three such that $e \subseteq \text{var}(C)$ for a clause $C \in F$. Thus STRONG \mathcal{C} -BACKDOOR SET DETECTION for $\mathcal{C} \in \{\text{HORN}, \text{HORN}^-, 2\text{CNF}\}$ can be accomplished by fpt algorithms for VERTEX COVER [19] and 3-HITTING SET [40]. The smallest strong 1-VAL-backdoor set of F is exactly the union of $\text{var}(C)$ for all negative clauses $C \in F$, the smallest strong 0-VAL-backdoor set of F is exactly the union of $\text{var}(C)$ for all positive clauses $C \in F$. \square

Proposition 6. STRONG RHORN-BACKDOOR SET DETECTION is W[2]-hard.

Proof. The proof uses a reduction from HS similar to the proof of Proposition 1. An instance $(\mathcal{S}, k), \mathcal{S} = \{S_1, \dots, S_m\}$, of HS is reduced to a formula F which is the union of certain gadgets G_i^j for $1 \leq i \leq m$ and $1 \leq j \leq k + 1$. Let $V = \bigcup_{i=1}^m S_i$. A gadget G_i^j contains the four clauses $S_i \cup \{z_1, z_2\}, \{z_1, \neg z_2\}, \{\neg z_1, z_2\}$, and $\overline{V} \cup \{\neg z_1, \neg z_2\}$, where z_1, z_2 are internal variables that do not occur outside the gadget. Let $B \subseteq V$ be a hitting set of \mathcal{S} and let $\tau \in 2^B$. If τ sets at least one variable to 0, then τ removes from each gadget the only negative clause, hence $r_{\text{var}(F)}(F[\tau]) \in \text{HORN}$. On the other hand, if τ sets all variables from B to 1, then it removes from each gadget the only positive clause (B is a hitting set). Hence, $F[\tau] \in \text{HORN}$ in this case. Consequently B is a strong RHORN-backdoor set of F . Conversely, assume B is a strong RHORN-backdoor set of F . Let $\tau \in 2^B$ be the all-1-assignment. For the sake of contradiction, assume there is a set S_i such that $B \cap S_i = \emptyset$. Since $|B| = k, B \cap \text{var}(G_i^j) = \emptyset$ for some $1 \leq j \leq k + 1$. Now $F[\tau]$ contains the subset $G_i^j[\tau] = \{S_i \cup \{z_1, z_2\}, \{z_1, \neg z_2\}, \{\neg z_1, z_2\}, \{\neg z_1, \neg z_2\}\}$ which is not renamable Horn, hence B is not a strong RHORN-backdoor set of F , a contradiction. Hence B is a hitting set of \mathcal{S} . \square

It is not known whether STRONG FOREST-BACKDOOR SET DETECTION is fixed-parameter tractable nor whether STRONG RHORN-BACKDOOR SET DETECTION is fixed-parameter tractable for 3CNF formulas. For the former problem, however, we know at least an fpt approximation [46]; see Theorem 5 below.

The following result is shown by a reduction from CYCLIC MONOTONE CIRCUIT ACTIVATION, similarly to Theorem 1.

Theorem 3 ([96]). STRONG \mathcal{C} -BACKDOOR SET DETECTION is $W[P]$ -complete for every base class $\mathcal{C} \in \text{Subsolver}$, even for formulas in 3CNF.

The bounded search tree method outlined above for WEAK CLU-BACKDOOR SET DETECTION for 3CNF formulas can clearly be adapted for strong backdoors. Hence we get the following result.

Proposition 7. STRONG CLU-BACKDOOR SET DETECTION is fixed-parameter tractable for 3CNF formulas.

5.1 Empty Clause Detection

Dilkina *et al.* [31] suggested to strengthen the concept of strong backdoor sets by means of *empty clause detection*. Let \mathcal{E} denote the class of all CNF formulas that contain the empty clause. For a base class \mathcal{C} we put $\mathcal{C}^{\{\emptyset\}} = \mathcal{C} \cup \mathcal{E}$; we call $\mathcal{C}^{\{\emptyset\}}$ the base class obtained from \mathcal{C} by adding empty clause detection. Formulas often have much smaller strong $\mathcal{C}^{\{\emptyset\}}$ -backdoor sets than strong \mathcal{C} -backdoor sets [31]. Dilkina *et al.* show that, given a CNF formula F and an integer k , determining whether F has a strong HORN $^{\{\emptyset\}}$ -backdoor set of size k , is both NP-hard and co-NP-hard (here k is considered just as part of the input and not as a parameter). Thus, the non-parameterized search problem for strong HORN-backdoor sets gets harder when empty clause detection is added. It turns out that also the parameterized problem gets harder when empty clause detection is added.

Theorem 4 ([97]). For every clause-induced base class \mathcal{C} such that at least one satisfiable CNF formula does not belong to \mathcal{C} the problem STRONG $\mathcal{C}^{\{\emptyset\}}$ -BACKDOOR SET is $W[1]$ -hard.

The theorem clearly applies to all base classes in $\text{Schaefer} \cup \{\text{RHORN}, \text{FOREST}\}$. The proof from [97] relies on a reduction from [39], where a reduction to 3CNF formulas is also given. Thus, Theorem 4 also holds for 3CNF formulas.

6 Detecting Deletion Backdoor Sets

In this section we consider the parameterized complexity of DELETION \mathcal{C} -BACKDOOR SET DETECTION for the various base classes \mathcal{C} from above. For most of the classes the complexity is easily established as follows. For Schaefer classes, strong and deletion backdoor sets coincide, hence the FPT results carry over. The subsolver classes are not clause-induced, hence it does not make sense to

consider deletion backdoor sets. DELETION FOREST-BACKDOOR SET DETECTION can be solved by algorithms for a slight variation of the feedback vertex set problem, and is therefore FPT. One has only to make sure that the feedback vertex set contains only variables and no clauses. This, however, can be achieved by using algorithms for WEIGHTED FEEDBACK VERTEX SET [81,17].

It is tempting to use Chen *et al.*'s FPT algorithm for directed feedback vertex set [20] for the detection of deletion backdoor sets. The corresponding base class would contain all CNF formulas with acyclic *directed* incidence graphs (the orientation of edges indicate whether a variable occurs positively or negatively). Unfortunately this class is not suited as a base class since it contains formulas where each clause contains either only positive literals or only negative literals, and SAT is well known to be NP-hard for such formulas [45].

Hence we are left with the classes CLU and RHORN.

For the detection of deletion CLU-backdoor sets we can use overlap obstructions and clash obstructions, as defined before Proposition 3. With each obstruction, we associate a deletion pair which is a pair of sets of variables. With an overlap obstruction $\{C_1, C_2\}$, we associate the deletion pair

$$\{\text{var}(C_1 \cap C_2), \text{var}((C_1 \setminus C_2) \cup (C_2 \setminus C_1))\},$$

and with a clash obstruction $\{D_1, D_2, D_3\}$, we associate the deletion pair

$$\{\text{var}((D_1 \setminus D_3) \cap \overline{D_2}), \text{var}((D_3 \setminus D_1) \cap \overline{D_2})\}.$$

For a formula F , let G_F denote the graph with vertex set $\text{var}(F)$ that has an edge xy if and only if there is a deletion pair $\{X, Y\}$ of F with $x \in X$ and $y \in Y$. Nishimura *et al.* [71] have shown that a set $X \subseteq \text{var}(F)$ is a deletion CLU-backdoor set of F if and only if X is a vertex cover of G_F . Thus, the detection of a deletion CLU-backdoor set of size k can be reduced to the problem of checking whether G_F has a vertex cover of size k , for which there exist very fast algorithms (see for example [19]).

Proposition 8 ([71]). DELETION CLU-BACKDOOR SET DETECTION is fixed-parameter tractable.

The remaining case is the class RHORN. As noted by Gottlob and Szeider [51] without proof (see also [82]), one can show fixed-parameter tractability of DELETION RHORN-BACKDOOR SET DETECTION by reducing it to the problem 2SAT DELETION. The latter problem takes as input a 2CNF formula and an integer k (the parameter), and asks whether one can make the formula satisfiable by deleting at most k clauses. 2SAT DELETION was shown fixed-parameter tractable by Razgon and O'Sullivan [82]. Here we give the above mentioned reduction.

Lemma 1. *There is a parameterized reduction from DELETION RHORN-BACKDOOR SET DETECTION to 2SAT DELETION.*

Proof. Let (F, k) be a given instance of DELETION RHORN-BACKDOOR SET DETECTION. We construct a graph $G = (V, E)$ by taking as vertices all literals

x^ϵ , for $x \in \text{var}(F)$ and $\epsilon \in \{0, 1\}$, and by adding two groups of edges. The first group consists of all edges x^0, x^1 for $x \in \text{var}(F)$, the second group consists of all edges $x^\epsilon y^\delta$ for $x, y \in \text{var}(F)$, $\epsilon, \delta \in \{0, 1\}$, such that $x^\epsilon, y^\delta \in C$ for some $C \in F$. Observe that the edges of the first group form a perfect matching M of the graph G .

Claim 1. F has a deletion RHORN-backdoor set of size at most k if and only if G has a vertex cover with at most $|M| + k$ vertices.

(\Rightarrow) Let B be a deletion RHORN-backdoor set of F of size at most k and $X \subseteq \text{var}(F) \setminus B$ such that $r_X(F - B) \in \text{HORN}$. Let $N = \{x^0 : x \in \text{var}(F) \setminus X\} \cup \{x^1 : x \in X\}$. Let $K = \{x^0, x^1 : x \in B\} \cup N$. By definition, $|K| = |M| + |B| \leq |M| + k$. We show that K is a vertex cover of G . Consider an edge $e = x^0 x^1 \in M$ of the first group. If $x \in X$, then $x^1 \in N \subseteq K$ and if $x \notin X$, then $x^0 \in N \subseteq K$. Hence e is covered by K . It remains to consider an edge $f = x^\epsilon y^\delta$ of the second group. If $x \in B$ or $y \in B$, then this edge is covered by K . Hence assume $x, y \notin B$. By construction of G , there is a clause $C \in F$ with $x^\epsilon, y^\delta \in C$. Since $x, y \notin B$, there is also a clause $C' \in F - B$ with $x^\epsilon, y^\delta \in C$. Since C' corresponds to a Horn clause $C'' \in r_X(F - B)$, at least one of the literals x^ϵ, y^δ belongs to N , and hence K covers the edge f . Hence the first direction of Claim 1 follows.

(\Leftarrow) Let K be a vertex cover of G with at most $|M| + k$ vertices. Let $B \subseteq \text{var}(F)$ be the set of all variables x such that both $x^0, x^1 \in K$. Clearly $|B| \leq k$. Let $X \subseteq \text{var}(F) \setminus B$ such that $x^1 \in K$. We show that $r_X(F - B) \in \text{HORN}$. Let x^δ, y^ϵ be two literals that belong to a clause C'' of $r_X(F - B)$. We show that $\epsilon = 0$ or $\delta = 0$. Let $C' \in F - B$ the clause that corresponds to C'' , and let $x^{\epsilon'}, y^{\delta'} \in C'$. It follows that $x^{\epsilon'} y^{\delta'} \in E$, and since K is a vertex cover of G , $x^{\epsilon'} \in K$ or $y^{\delta'} \in K$. If $x^{\epsilon'} \in K$ then $\epsilon = 0$, if $y^{\delta'} \in K$ then $\delta = 0$. Since $x^\delta, y^\epsilon \in C'' \in r_X(F - B)$ were chosen arbitrarily, we conclude that $r_X(F - B) \in \text{HORN}$. Hence Claim 1 is shown.

Mishra *et al.* [68] already observed that a reduction from [16] can be adapted to show that this above-guarantee vertex cover problem can be reduced to 2SAT DELETION. For completeness, we give a reduction here as well.

We construct a 2CNF formula F_2 from G . For each vertex x^ϵ of G we take a variable x_ϵ . For each edge $x^0 x^1 \in M$ we add a negative clause $\{\neg x_0, \neg x_1\}$, and for each edge $x^\epsilon y^\delta \in E \setminus M$ we add a positive clause $\{x_\epsilon, y_\delta\}$.

Claim 2. G has a vertex cover with at most $|M| + k$ vertices if and only if we can delete at most k negative clauses from F_2 to obtain a satisfiable formula.

(\Rightarrow) Let K be a vertex cover of G . We delete from F_2 all negative clauses $\{\neg x_0, \neg x_1\}$ where both $x_0, x_1 \in K$ (there are at most k such clauses) and obtain a 2CNF formula F'_2 . We define a truth assignment $\tau \in 2^{\text{var}(F'_2)}$ by setting a variable to 1 if and only if it belongs to K . It remains to show that τ satisfies F'_2 . The negative clauses are satisfied since τ sets exactly one literal of a negative clause $\{\neg x_0, \neg x_1\} \in F'_2$ to 1 and exactly one to 0. The positive clauses are satisfied since each positive clause $\{x_\epsilon, y_\delta\}$ corresponds to an edge $x_\epsilon y_\delta \in E$, and since K is a vertex cover, τ sets at least one of the variables x_ϵ, y_δ to 1.

(\Leftarrow) Let F'_2 be a satisfiable formula obtained from F_2 by deleting at most k negative clauses. Let $D = \{x \in \text{var}(F) : \{\neg x_0, \neg x_1\} \in F_2 \setminus F'_2\}$. Let τ

be a satisfying truth assignment of F'_2 . We define a set K of vertices of G by setting $K = \{x^0, x^1 : x \in D\} \cup \{x^{\tau(x)} : x \in \text{var}(F) \setminus D\}$, and we observe that $|K| \leq |M| + k$. It remains to show that K is a vertex cover of G . Consider an edge $e = x^0x^1 \in M$ of the first group. If $x \in D$ then $x^0, x^1 \in K$; if $x \notin D$ then $x^{\tau(x)} \in K$, hence e is covered by K . Now consider an edge $f = x^\epsilon y^\delta \in E \setminus M$ of the second group. If $x \in D$ or $y \in D$ then f is clearly covered by K . Hence assume $x, y \notin D$. By definition, there is a positive clause $\{x_\epsilon, y_\delta\} \in F'_2 \subseteq F_2$. Since τ satisfies F'_2 , it follows that $\tau(x_\epsilon) = 1$ or $\tau(y_\delta) = 1$. Consequently $x^\epsilon \in K$ or $y^\delta \in K$, thus K covers f . Hence Claim 2 is shown.

Next we modify F_2 by replacing each positive clause $C = \{x_\epsilon, y_\delta\}$ with $2k + 2$ ‘‘mixed’’ clauses $\{x_\epsilon, z^i_C\}, \{\neg z^i_C, y_\delta\}$, for $1 \leq i \leq k + 1$, where the z^i_C are new variables. Let F_2^* denote the 2CNF formula obtained this way from F_2 .

Claim 3. We can delete at most k negative clauses from F_2 to obtain a satisfiable formula if and only if we can delete at most k clauses from F_2^* to obtain a satisfiable formula.

The claim follows easily from the following considerations. We observe that each pair of mixed clauses $\{x_\epsilon, z^i_C\}, \{\neg z^i_C, y_\delta\}$ is semantically equivalent with $C = \{x_\epsilon, y_\delta\}$. Hence, if F_2 can be made satisfiable by deleting some of the negative clauses, we can also make F_2^* satisfiable by deleting the same clauses. However, deleting some of the mixed clauses does only help if we delete at least one from each of the $k + 1$ pairs that correspond to the same clause C . Hence also Claim 3 is shown true. Claims 1–3 together establish the lemma. \square

Razgon and O’Sullivan’s result [82] together with Lemma 1 immediately give the following.

Proposition 9. DELETION RHORN-BACKDOOR SET DETECTION is fixed-parameter tractable.

7 Permissive Problems

We consider any function p that assigns nonnegative integers to CNF formulas as a *satisfiability parameter*. In particular we are interested in such satisfiability parameters p for which the following parameterized problem is fixed-parameter tractable:

SAT(p)

Instance: A CNF formula F and an integer $k \geq 0$.

Parameter: The integer k .

Task: Determine whether F is satisfiable or determine that $p(F) > k$.

Note that an algorithm that solves the problem has the freedom of deciding the satisfiability of some formulas F with $p(F) > k$, hence the exact recognition of formulas F with $p(F) \leq k$ can be avoided. Thus SAT(p) is not a usual decision problem, as there are three different outputs, not just two. If SAT(p) is fixed-parameter tractable then we call p an fpt satisfiability parameter, and we say that ‘‘the satisfiability of CNF formulas of bounded p is fixed-parameter tractable’’ (cf. [95]). We write 3SAT(p) if the input is restricted to 3CNF formulas.

Backdoor sets provide a generic way to define satisfiability parameters. Let \mathcal{C} be a base class and F a CNF formula. We define $\text{wb}_{\mathcal{C}}(F)$, $\text{sb}_{\mathcal{C}}(F)$ and $\text{db}_{\mathcal{C}}(F)$ as the size of a smallest weak, strong, and deletion \mathcal{C} -backdoor set of F , respectively.

Of course, if the detection of the respective \mathcal{C} -backdoor set is fixed-parameter tractable, then $\text{wb}_{\mathcal{C}}$, $\text{sb}_{\mathcal{C}}$, and $\text{db}_{\mathcal{C}}$ are fpt satisfiability parameters. However, it is possible that $\text{wb}_{\mathcal{C}}$, $\text{sb}_{\mathcal{C}}$, or $\text{db}_{\mathcal{C}}$ are fpt satisfiability parameters but the corresponding \mathcal{C} -backdoor set detection problem is $W[1]$ -hard. The problems $\text{SAT}(\text{wb}_{\mathcal{C}})$, $\text{SAT}(\text{sb}_{\mathcal{C}})$, and $\text{SAT}(\text{db}_{\mathcal{C}})$ can therefore be considered as more “permissive” versions of the “strict” problems WEAK, STRONG, and DELETION \mathcal{C} -BACKDOOR SET DETECTION, the latter require to find a backdoor set even if the given formula is trivially seen to be satisfiable or unsatisfiable. The distinction between permissive and strict versions of problems have been considered in a related context by Marx and Schlotter [66,67] for parameterized k -neighborhood local search. Showing hardness for permissive problems $\text{SAT}(p)$ seems to be a much more difficult task than for the strict problems. So far we could establish only few such hardness results.

Proposition 10. $\text{SAT}(\text{wb}_{\mathcal{C}})$ is $W[1]$ -hard for all $\mathcal{C} \in \text{Schaefer} \cup \{\text{RHORN}\}$.

Proof. We will show a more general result, that $W[1]$ -hardness holds for all base classes that contain all anti-monotone 2CNF formulas. A CNF formula is *anti-monotone* if all its clauses are negative. Let \mathcal{C} be a base class that contains all anti-monotone 2CNF formulas.

We show that $\text{SAT}(\text{wb}_{\mathcal{C}})$ is $W[1]$ -hard by reducing from PARTITIONED CLIQUE, also known as MULTICOLORED CLIQUE. This problem takes as input a k -partite graph and asks whether the graph has a clique on k vertices. The integer k is the parameter. The problem is well-known to be $W[1]$ -complete [78].

Let $H = (V, E)$ with $V = \bigcup_{i=1}^k V_i$ be an instance of this problem. We construct a CNF formula F as follows. We consider the vertices of H as variables and add clauses $\{\neg u, \neg v\}$ for any two distinct vertices such that $uv \notin E$. For each $1 \leq i \leq k$, we add the clause V_i . This completes the construction of F .

We show that the following statements are equivalent:

- (1) F is satisfiable
- (2) H contains a k -clique.
- (3) F has a weak \mathcal{C} -backdoor set of size at most k .

(1) \Rightarrow (2). Let τ be a satisfying assignment of F . Because of the clause V_i , τ sets at least one variable of V_i to 1, for each $1 \leq i \leq k$. As each V_i is an independent set, F contains a clause $\{\neg u, \neg v\}$ for every two distinct vertices in V_i . Thus, τ sets exactly one variable of V_i to 1, for each $1 \leq i \leq k$. The clauses of F also imply that $v_i v_j \in E$ for each $1 \leq i < j \leq k$, since otherwise τ would falsify the clause $\{\neg v_i, \neg v_j\}$. Hence v_1, \dots, v_k induce a clique in H .

(2) \Rightarrow (3). Assume v_1, \dots, v_k induce a clique in H , with $v_i \in V_i$. We show that $B = \{v_1, \dots, v_k\}$ is a weak \mathcal{C} -backdoor set of F . Let $\tau \in 2^B$ be the truth assignment that sets all variables of B to 1. This satisfies all the clauses V_i , $1 \leq i \leq k$. Thus, $F[\tau]$ is an anti-monotone 2CNF formula. Therefore it is in \mathcal{C} and it is satisfiable as it is 0-valid. Hence B is a weak \mathcal{C} -backdoor set of F .

(3) \Rightarrow (1). Any formula that has a weak backdoor set is satisfiable.

Since all three statements are equivalent, we conclude that $\text{SAT}(\text{wb}_{\mathcal{C}})$ is $\text{W}[1]$ -hard. This shows the proposition for the base classes HORN, 2CNF, 0-VAL, and RHORN, as they contain all anti-monotone 2CNF formulas. The hardness for HORN^- and 1-VAL follows by symmetric arguments from the hardness of HORN and 0-VAL, respectively. \square

In general, if we have an *fpt approximation algorithm* [14,21,34] for a strict backdoor set detection problem, then the corresponding permissive problem $\text{SAT}(p)$ is fixed-parameter tractable. For instance, if we have an fpt algorithm that, for a given pair (F, k) either outputs a weak, strong, or deletion \mathcal{C} -backdoor set of F of size at most $f(k)$ or decides that F has no such backdoor set of size at most k , then clearly $\text{wb}_{\mathcal{C}}$, $\text{sb}_{\mathcal{C}}$, and $\text{db}_{\mathcal{C}}$, respectively, is an fpt satisfiability parameter.

This line of reasoning is used in the next theorem to show that $\text{sb}_{\text{FOREST}}$ is an fpt satisfiability parameter. This result labels FOREST as the first nontrivial base class \mathcal{C} for which $\text{sb}_{\mathcal{C}}$ is an fpt satisfiability parameter and $\text{sb}_{\mathcal{C}} \neq \text{db}_{\mathcal{C}}$. Hence the additional power of strong FOREST-backdoor sets over deletion FOREST-backdoor sets is accessible.

Theorem 5 ([46]). STRONG FOREST-BACKDOOR SET DETECTION admits a 2^k fpt-approximation. Hence $\text{SAT}(\text{sb}_{\text{FOREST}})$ is fixed-parameter tractable.

Proof (Sketch). We sketch the fpt-approximation algorithm from [46] which either concludes that a CNF formula F has no strong FOREST-backdoor set of size k or returns one of size at most 2^k . We refer to [46] for the full details and the correctness proof. Let G denote the incidence graph of F . The first step of the algorithm runs, similarly to the proof of Theorem 2, the fpt algorithm (with parameter k') by Bodlaender [9] that either finds $k' = k^2 2^{k-1} + k + 1$ vertex-disjoint cycles in G or a feedback vertex set of G of size at most $12k'^2 - 27k' + 15$.

In case a feedback vertex set X is returned, a tree decomposition of $G \setminus X$ of width 1 is computed and X is added to each bag of this tree decomposition. As the STRONG FOREST-BACKDOOR SET DETECTION problem can be defined in Monadic Second Order Logic, a meta-theorem by Courcelle [26] can be used to decide the problem in linear time using this tree decomposition.

In case Bodlaender’s algorithm returns k' vertex-disjoint cycles, the algorithm finds a set S^* of $O(k^{2k} 2^{k^2 - k})$ variables such that every strong FOREST-backdoor set of size k contains at least one variable from S^* . In this case, the algorithm recurses by considering all possibilities of including a variable from S^* in the backdoor set.

Let $C_1, \dots, C_{k'}$ denote the variable-disjoint cycles returned by Bodlaender’s algorithm. Consider a variable $x \in \text{var}(F)$ and a cycle C . We say that x kills C internally if $x \in C$. We say that x kills C externally if $x \notin C$ and C contains two clause $u, v \in F$ such that $x \in u$ and $\neg x \in v$. We say in this case that x kills C externally in u and v .

The algorithm goes through all $\binom{k'}{k}$ ways to choose k cycles among $C_1, \dots, C_{k'}$ that may be killed internally. All other cycles, say $C_1, \dots, C_{k''}$ with $k'' = k' - k$,

are not killed internally. We refer to these cycles as C'' -cycles. The algorithm now computes a set $S \subseteq \text{var}(F)$ of size at most 2 such that any strong FOREST-backdoor set of size k , which is a subset of $\text{var}(F) \setminus \bigcup_{i=1}^{k''} \text{var}(C_i)$, contains at least one variable from S . The union of all such S , taken over all choices of cycles to be killed internally, forms then the set S^* that was to be computed.

From now on, consider only killers in $\text{var}(F) \setminus \bigcup_{i=1}^{k''} \text{var}(C_i)$. For each C'' -cycle C_i , consider vertices x_i, u_i, v_i such that x_i kills C_i externally in u_i and v_i and there is a path P_i from u_i to v_i along the cycle C_i such that if any variable kills C_i externally in two clauses u'_i and v'_i such that $u'_i, v'_i \in P_i$, then $\{u_i, v_i\} = \{u'_i, v'_i\}$. Note that any variable that does not kill C_i internally, but kills the cycle $Cx_i = P_i \cup \{x_i\}$ also kills the cycle C_i externally in u_i and v_i . We refer to such external killers as *interesting*.

The algorithm executes the first applicable from the following rules.

- No External Killer.** If there is an index $i, 1 \leq i \leq k''$, such that Cx_i has no external killer, then set $S := \{x_i\}$.
- Killing Same Cycles.** If there are variables y and z and at least $2^{k-1} + 1$ C'' -cycles such that both y and z are interesting external killers of each of these C'' -cycles, then set $S := \{y, z\}$.
- Killing Many Cycles.** If there is a variable y that is an interesting external killer of at least $k \cdot 2^{k-1} + 1$ C'' -cycles, then set $S := \{y\}$.
- Too Many Cycles** Otherwise, set $S = \emptyset$.

For each $s \in S^*$ the algorithm calls itself recursively to compute a strong FOREST-backdoor set for $F[s = 0]$ and for $F[s = 1]$ with parameter $k - 1$. If both recursive calls return backdoor sets, the union of these two backdoor sets and $\{s\}$ is a strong FOREST-backdoor set for F . It returns the smallest such backdoor set obtained for all choices of s , or NO if for each $s \in S^*$ at least one recursive call returned NO. □

Very recently, Theorem 5 has been extended to the base classes NESTED [47] and $\mathcal{W}_{\leq t}$, for every fixed $t \geq 0$ [48]. The class NESTED was introduced by Knuth [59]. It is the class of all CNF formulas whose variables can be linearly ordered such that no pair of clauses straddle each other; a clause c *straddles* a clause c' if there are variables $x, y \in \text{var}(c)$ and $z \in \text{var}(c')$ such that $x < z < y$ in the linear ordering under consideration. The class $\mathcal{W}_{\leq t}$ contains all CNF formulas whose incidence graph has treewidth at most t . These results generalize Theorem 5 since $\mathcal{W}_{\leq 1} = \text{FOREST} \subseteq \text{NESTED} \subseteq \mathcal{W}_{\leq 3}$. The overall outline of the algorithms from [47,48] resembles the algorithm presented in the proof of Theorem 5, but the case where the incidence graph has large treewidth requires significantly more involved arguments.

8 Comparison of Parameters

Satisfiability parameters can be compared with respect to their generality. Let p, q be satisfiability parameters. We say that p is *at least as general* as q , in

symbols $p \preceq q$, if there exists a function f such that for every CNF formula F we have $p(F) \leq f(q(F))$. Clearly, if $p \preceq q$ and $\text{SAT}(p)$ is fpt, then so is $\text{SAT}(q)$. If $p \preceq q$ but not $q \preceq p$, then p is *more general* than q . If neither $p \preceq q$ nor $q \preceq p$ then p and q are *incomparable*.

As discussed above, each base class \mathcal{C} gives rise to three satisfiability parameters $\text{wb}_{\mathcal{C}}(F)$, $\text{sb}_{\mathcal{C}}(F)$ and $\text{db}_{\mathcal{C}}(F)$. If \mathcal{C} is clause-induced, then $\text{sb}_{\mathcal{C}} \preceq \text{db}_{\mathcal{C}}$; and if $\mathcal{C} \subseteq \mathcal{C}'$, then $\text{sb}_{\mathcal{C}'} \preceq \text{sb}_{\mathcal{C}}$ and $\text{db}_{\mathcal{C}'} \preceq \text{db}_{\mathcal{C}}$.

By associating certain graphs with CNF formulas one can use graph parameters to define satisfiability parameters. The most commonly used graphs are the primal, dual, and incidence graphs. The primal graph of a CNF formula F has as vertices the variables of F , and two variables are adjacent if they appear together in a clause. The dual graph has as vertices the clauses of F , and two clauses C, C' are adjacent if they have a variable in common (i.e., if $\text{var}(C) \cap \text{var}(C') \neq \emptyset$). The incidence graph, as already defined above, is a bipartite graph, having as vertices the variables and the clauses of F ; a variable x and a clause C are adjacent if $x \in \text{var}(C)$. The directed incidence graph is obtained from the incidence graph by directing an edge xC from x to C if $x \in C$ and from C to x if $\neg x \in C$.

The treewidth of the primal, dual, and incidence graph gives fpt satisfiability parameters, respectively. The treewidth of the incidence graph is more general than the other two satisfiability parameters [60]. The clique-width of the three graphs provides three more general satisfiability parameters. However, these satisfiability parameters are unlikely fpt: It is easy to see that SAT remains NP-hard for CNF formulas whose primal graphs are cliques, and for CNF formulas whose dual graphs are cliques. Moreover, SAT, parameterized by the clique-width of the incidence graph is W[1]-hard, even if a decomposition is provided [72]. However, the clique-width of directed incidence graphs is an fpt satisfiability parameter which is more general than the treewidth of incidence graphs [25,43].

How do fpt satisfiability parameters based on decompositions and fpt satisfiability parameters based on backdoor sets compare to each other?

Each base class \mathcal{C} considered above, except for the class FOREST, contains CNF formulas whose directed incidence graphs have arbitrarily large clique-width. Hence none of the decomposition based parameters is at least as general as the parameters $\text{sb}_{\mathcal{C}}$ and $\text{db}_{\mathcal{C}}$. On the other hand, taking the disjoint union of n copies of a CNF formula multiplies the size of backdoor sets by n but does not increase the width. Hence no backdoor based parameter is more general than decomposition based parameters.

Thus, almost all considered backdoor based fpt satisfiability parameters are incomparable with almost all considered decomposition based fpt satisfiability parameters. A notable exception is the satisfiability parameter $\text{db}_{\text{FOREST}}$. It is easy to see that the treewidth of the incidence graph of a CNF formula is no greater than the size of a smallest deletion FOREST-backdoor set plus one, as the latter forms a feedback vertex set of the incidence graph. Thus the treewidth of incidence graphs is a more general satisfiability parameter than the size of a smallest deletion FOREST-backdoor sets. However, one can construct CNF formulas F with $\text{sb}_{\text{FOREST}}(F) = 1$ whose directed incidence graph has arbitrarily

large clique-width. Just take a formula whose incidence graph is a subdivision of a large square grid, and add a further variable x such that on each path which is a subdivision of one edge of the grid there is a clause containing x and a clause containing $\neg x$. Thus, the satisfiability parameter $\text{sb}_{\text{FOREST}}$, which is fpt by Theorem 5, is incomparable to all the decomposition based satisfiability parameters considered above.

Figure 1 shows the relationship between some of the discussed fpt satisfiability parameters.

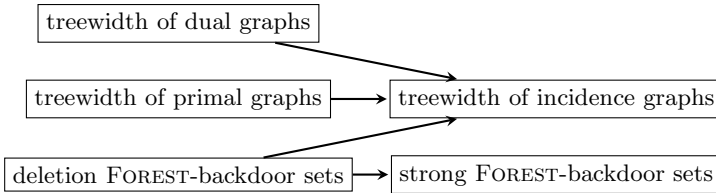


Fig. 1. Relationship between some fpt satisfiability parameters. An arrow from A to B means that B is more general than A . If there is now arrow between A and B then A and B are incomparable.

9 Kernels

The use of strong or deletion backdoor sets for SAT decision, with respect to a base class \mathcal{C} , involves two tasks:

1. *backdoor detection*, to find a strong (or deletion) backdoor set of size at most k , or to report that such a backdoor set does not exist,
2. *backdoor evaluation*, to use a given strong (or deletion) backdoor set of size at most k to determine whether the CNF formula under consideration is satisfiable.

In each case where backdoor detection is fixed-parameter tractable, one can now ask whether the detection problem admits a polynomial kernel. For instance, for the classes HORN and 2CNF, backdoor detection can be rephrased as VERTEX COVER or as 3-HITTING SET problems, as discussed above, and therefore admits polynomial kernels [18,1].

Backdoor evaluation is trivially fixed-parameter tractable for any base class, but it is unlikely that it admits a polynomial kernel.

Proposition 11 ([98]). \mathcal{C} -BACKDOOR SET EVALUATION does not admit a polynomial kernel for any self-reducible base class \mathcal{C} unless $\text{NP} \subseteq \text{co-NP/poly}$.

This proposition is a trivial consequence of the well-known result that SAT parameterized by the number of variables has no polynomial kernel unless $\text{NP} \subseteq \text{co-NP/poly}$ [10,44], and the fact that $\text{var}(F)$ is always a strong \mathcal{C} -backdoor set of F if \mathcal{C} is self-reducible.

Less immediate is the question whether \mathcal{C} -BACKDOOR SET EVALUATION admits a polynomial kernel if the inputs are restricted to 3CNF formulas, as 3SAT parameterized by the number of variables has a cubic kernel by trivial reasons. However, for HORN and 2CNF this question can be answered negatively.

Proposition 12 ([98]). *\mathcal{C} -BACKDOOR SET EVALUATION does not admit a polynomial kernel for $\mathcal{C} \in \{\text{HORN}, 2\text{CNF}\}$ unless $\text{NP} \subseteq \text{co-NP/poly}$, even if the input formula is in 3CNF.*

10 Backdoor Trees

Backdoor trees are binary decision trees on backdoor variables whose leaves correspond to instances of the base class. Every strong backdoor set of size k gives rise to a backdoor tree with at least $k + 1$ and at most 2^k leaves. It is reasonable to rank the hardness of instances in terms of the number of leaves of backdoor trees, thus gaining a more refined view than by just comparing the size of backdoor sets.

Consider the CNF formula F with variables x_1, \dots, x_{2n} and y_1, \dots, y_n consisting of all clauses of the form

$$\begin{aligned} &\{y_i, \neg x_1, \dots, \neg x_{2i-2}, x_{2i-1}, \neg x_{2i}, \dots, \neg x_{2n}\}, \\ &\{y_i, \neg x_1, \dots, \neg x_{2i-1}, x_{2i}, \neg x_{2i+1}, \dots, \neg x_{2n}\}, \end{aligned}$$

for $1 \leq i \leq n$. The set $B = \{y_1, \dots, y_n\}$ is a strong HORN-backdoor set (in fact, B is the smallest possible). However, every HORN-backdoor tree T with $\text{var}(T) = \{y_1, \dots, y_n\}$ has 2^n leaves. On the other hand, the formula F has a HORN-backdoor tree T' with only $2n + 1$ leaves where $\text{var}(T') = \{x_1, \dots, x_{2n}\}$. Thus, when we want to minimize the number of leaves of backdoor trees, we must not restrict ourselves to variables of a smallest strong backdoor set.

The problem \mathcal{C} -BACKDOOR TREE DETECTION now takes as input a CNF formula F , a parameter k , and asks whether F has a \mathcal{C} -backdoor tree with at most k leaves.

A base class \mathcal{C} is said to admit a *loss-free kernelization* if there exists a polynomial-time algorithm that, given a CNF formula F and an integer k , either correctly decides that F has no strong \mathcal{C} -backdoor set of size at most k , or computes a set $X \subseteq \text{var}(F)$ such that the following conditions hold: (i) X contains all minimal strong \mathcal{C} -backdoor sets of F of size at most k ; and (ii) the size of X is bounded by a computable function that depends on k only.

Samer and Szeider [88] have shown that \mathcal{C} -BACKDOOR TREE DETECTION is fixed-parameter tractable for every base class \mathcal{C} that admits a loss-free kernelization. Since Buss-type kernelization is loss-free, the two classes HORN and 2CNF admit a loss-free kernelization. Hence \mathcal{C} -BACKDOOR TREE DETECTION is fixed-parameter tractable for $\mathcal{C} \in \{2\text{CNF}, \text{HORN}\}$.

11 Backdoors for Problems beyond NP

The backdoor approach has been successfully applied to obtain fixed-parameter tractability for problems whose unparameterized worst-case complexity lies

beyond NP. In particular, FPT results have been obtained for the $\#P$ -complete problem Propositional Model Counting, the PSPACE-complete QBF-SAT problem, and problems of nonmonotonic reasoning and abstract argumentation that are located on the second level of the Polynomial Hierarchy. In this section we briefly survey these results.

11.1 Propositional Model Counting

The $\#SAT$ problem asks to compute for a given CNF formula F the number of assignments $\tau \in 2^{\text{var}(F)}$ that satisfy F . This problem arises in several areas of Artificial Intelligence, in particular in the context of probabilistic reasoning [3,86]. The problem is $\#P$ -complete and remains $\#P$ -hard even for monotone 2CNF formulas and Horn 2CNF formulas. It is NP-hard to approximate the number of satisfying assignments of a CNF formula with n variables within $2^{n^{1-\epsilon}}$ for any $\epsilon > 0$. This approximation hardness holds also for monotone 2CNF formulas and Horn 2CNF formulas [86]. However, if $\#SAT$ can be solved in polynomial time $O(n^c)$ for the formulas of a base class \mathcal{C} , and if we know a strong \mathcal{C} -backdoor set of a formula F of size k , then we can compute the number of satisfying assignments of F in time $O(2^k n^c)$ [71,90]. For some applications in probabilistic reasoning one is interested in the weighted model counting (WMC) problem, which is more general than $\#SAT$ (see, e.g., [92,15]). Since the backdoor set approach applies also to the more general problem, we will use it for the following discussions.

A *weighting* w of a CNF formula F is a mapping w that assigns each variable $x \in \text{var}(F)$ a rational number $0 \leq w(x) \leq 1$; this generalizes to literals by $w(\bar{x}) = 1 - w(x)$ and to truth assignments $\tau \in 2^X$ by $w(\tau) = \prod_{x \in X} w(x^{\tau(x)})$. We define $\#_w(F)$ as the sum of the weights of all assignments $\tau \in 2^{\text{var}(F)}$ that satisfy F . The WMC problem asks to compute $\#_w(F)$ for a given CNF formula F and weighting w . WMC is clearly at least as hard as computing $\#(F)$ as we can reduce $\#SAT$ to WMC by using the weight $1/2$ for all n variables and multiplying the result by 2^n . A strong \mathcal{C} -backdoor set X of a CNF formula F can be used to compute $\#_w(F)$ via the equation

$$\#_w(F) = \sum_{\tau \in 2^X} w(\tau) \cdot \#_w(F[\tau]).$$

It is easy to see that WMC is polynomial for the base classes CLU and FOREST as the corresponding algorithms for deciding satisfiability for these classes as discussed above allow a straightforward generalization to WMC. From Theorem 5 and Proposition 8 we conclude that WMC is fixed-parameter tractable parameterized by $\text{sb}_{\text{FOREST}}$ and db_{CLU} .

11.2 Quantified Boolean Formulas

Many important computational tasks like planning, verification, and several questions of knowledge representation and automated reasoning can be

naturally encoded as the evaluation problem of *quantified Boolean formulas (QBF)* [74,84,87]. A QBF consists of a propositional CNF formula F (the “matrix”) and a quantifier prefix. For instance $\mathcal{F} = \forall y \forall z \exists x \exists w F$ with $F = \{\{\neg x, y, \neg w\}, \{x, \neg y, w\}, \{\neg y, z\}, \{y, \neg z\}\}$ is a QBF. The evaluation of quantified Boolean formulas constitutes a PSPACE-complete problem and is therefore believed to be computationally harder than the NP-complete propositional satisfiability problem [58,76,94]. Only a few tractable classes of quantified Boolean formulas are known where the number of *quantifier alternations* is unbounded. For example, the time needed to solve QBF formulas whose primal graph has bounded treewidth grows non-elementarily in the number of quantifier alternations [75]. Two prominent tractable classes with *unbounded* quantifier alternations are QHORN and Q2CNF which are QBFs where the matrix is a Horn or 2CNF formula, respectively. QHORN formulas and Q2CNF formulas can be evaluated in polynomial time due to well-known results of Kleine Büning *et al.* [13] and of Aspvall *et al.* [2], respectively.

In order to evaluate a QBF formula with a small strong HORN- or 2CNF-backdoor set X efficiently, we require that X is closed under variable dependencies. That is, if x depends on y and $x \in X$, then also $y \in X$, where we say that x depends on y if the quantifier for y appears to the left of the quantifier for x , and one cannot move the quantifier for y to the right of x without changing the validity of the QBF. In general deciding whether a variable depends on the other is PSPACE complete, but there are “over-approximations” of dependencies that can be computed in polynomial time. Such over-approximations can be formalized in terms of *dependency schemes*. Indeed, it is fixed-parameter tractable to detect strong HORN or 2CNF-backdoor sets of size at most k that are closed with respect to any fixed polynomial-time decidable dependency scheme [89]. This fpt result allows an unbounded number of quantifier alternations for each value of the parameter, in contrast to the results for parameter treewidth.

11.3 Nonmonotonic Reasoning

Answer-Set Programming (ASP) is an increasingly popular framework for declarative programming [65,69]. ASP allows to describe a problem by means of rules and constraints that form a disjunctive logic program P over a finite universe U of atoms. A rule r is of the form $(x_1 \vee \dots \vee x_l \leftarrow y_1, \dots, y_n, \neg z_1, \dots, \neg z_m)$. We write $\{x_1, \dots, x_l\} = H(r)$ (the *head* of r) and $\{y_1, \dots, y_n, z_1, \dots, z_m\} = B(r)$ (the *body* of r), $B^+(r) = \{y_1, \dots, y_n\}$ and $B^-(r) = \{z_1, \dots, z_m\}$. A set M of atoms *satisfies* a rule r if $B^+(r) \subseteq M$ and $M \cap B^-(r) = \emptyset$ implies $M \cap H(r) \neq \emptyset$. M is a *model* of P if it satisfies all rules of P . The *GL reduct* of a program P under a set M of atoms is the program P^M obtained from P by first removing all rules r with $B^-(r) \cap M \neq \emptyset$ and second removing all $\neg z$ where $z \in B^-(r)$ from all remaining rules r [49]. M is an *answer set* of a program P if M is a minimal model of P^M .

For instance, from the program $P = \{(\text{tweety-flies} \leftarrow \text{tweety-is-a-bird}, \neg \text{tweety-is-a-penguin}), (\text{tweety-is-a-bird} \leftarrow)\}$ we may conclude that *tweety-flies*, since this fact is contained in the only answer set $\{\text{tweety-is-a-bird}, \text{tweety-flies}\}$

of P . If we add the fact *tweety-is-a-penguin* to the program and obtain $P' = P \cup \{(\text{tweety-is-a-penguin} \leftarrow)\}$, then we have to retract our conclusion *tweety-flies* since this fact is not contained in any answer set of P' (the only answer set of P' is $\{\text{tweety-is-a-bird}, \text{tweety-is-a-penguin}\}$). This *nonmonotonic* behaviour that adding a fact may allow fewer conclusions is typical for many applications in Artificial Intelligence. The main computational problems for ASP (such as deciding whether a program has a solution, or if a certain atom is contained in at least one or in all answer sets) are of high worst-case complexity and are located at the second level of the Polynomial Hierarchy [38].

Also for ASP several islands of tractability are known, and it is possible to develop a backdoor approach [42]. Similar to SAT one can define partial truth assignments τ on a set of atoms and solve a disjunctive logic program P by solving all the reduced programs $P[\tau]$. However, the situation is trickier than for satisfiability. Although every answer set of P corresponds to an answer set of $P[\tau]$ for some truth assignment τ , the reverse direction is not true. Therefore, one needs to run a check for each answer set of $P[\tau]$ whether it gives rise to an answer set of P . Although this correctness check is polynomial, we must ensure that we do not need to carry it out too often. A sufficient condition for bounding the number of checks is that we can compute all answer sets of a program $P \in \mathcal{C}$ in polynomial time (" \mathcal{C} is enumerable"). In particular, this means that $P \in \mathcal{C}$ has only a polynomial number of answer sets, and so we need to run the correctness check only a polynomial number of times.

Several enumerable islands of tractability have been identified and studied regarding the parameterized complexity of backdoor set detection [42]. For instance, programs where each rule head contains exactly one atom and each rule body is negation-free are well-known to have exactly one answer set. Such programs are called Horn programs, and similar to satisfiability, one can use vertex covers to compute backdoor sets with respect to Horn. Further enumerable islands of tractability can be defined by forbidding cycles in graphs, digraphs, and mixed graphs associated with disjunctive logic programs. Now, one can use feedback vertex set (fvs) algorithms for the considered graphs to compute backdoor sets: undirected fvs [33], directed fvs [20], and mixed fvs [12]. One can get even larger enumerable islands of tractability by labeling some of the vertices or edges and by only forbidding "bad" cycles, namely cycles that contain at least one labeled edge or vertex. For the undirected case one can use subset feedback vertex set algorithms to compute backdoor sets [28,57]. Currently it is open whether this problem is fixed-parameter tractable for directed or mixed graphs. Even larger islands can be obtained by only forbidding bad cycles with an even number of labeled vertices or edges [41]. This gives rise to further challenging feedback vertex set problems.

11.4 Abstract Argumentation

The study of arguments as abstract entities and their interaction in form of *attacks* as introduced by Dung [35] has become one of the most active research branches within Artificial Intelligence, Logic and Reasoning [5,7,80]. Abstract

argumentation provides suitable concepts and formalisms to study, represent, and process various reasoning problems most prominently in defeasible reasoning (see, e.g., [79,11]) and agent interaction (see, e.g., [77]). An abstract argumentation system can be considered as a directed graph, where the vertices are called “arguments” and a directed edge from a to b means that argument a “attacks” argument b .

A main issue for any argumentation system is the selection of acceptable sets of arguments, called extensions. Whether or not a set of arguments is accepted is considered with respect to certain properties of sets of arguments, called semantics [4]. For instance, the *preferred semantics* requires that an extension is a maximal set of arguments with the properties that (i) the set is independent, and (ii) each argument outside the set which attacks some argument in the set is itself attacked by some argument in the set. Property (i) ensures that the set is conflict-free, property (ii) ensures that the set defends itself against attacks.

Important computational problems are to determine whether an argument belongs to some extension (credulous acceptance) or whether it belongs to all extensions (skeptical acceptance) [32,37]. For most semantics, including the preferred semantics, the problems are located on the second level of the Polynomial Hierarchy [36].

It is known that the acceptance problems can be solved in polynomial time if the directed graph of the argumentation framework is *acyclic*, *noeven* (contains no even cycles), *symmetric*, or *bipartite* [35,4,24,36]. Thus, these four properties give rise to islands of tractability for abstract argumentation, and one can ask whether a backdoor approach can be developed to solve the acceptance problems for instances that are close to an island. Here it is natural to consider deletion backdoor sets, i.e., we delete arguments to obtain an instance that belongs to the considered class. For the islands of acyclic, symmetric, and bipartite argumentation frameworks we can find a backdoor using the fixed-parameter algorithms for directed feedback vertex set [20], vertex cover [33] and for graph bipartization [83], respectively. For finding a vertex set of size k that kills all directed cycles of even length we only know an XP algorithm which is based on a deep result [85].

However, it turns out that using the backdoor set is tricky and quite different from satisfiability and answer set programming [73]. The acceptance problems remain (co-)NP-hard for instances that can be made symmetric or bipartite by deleting one single argument. On the other hand, if an instance can be made acyclic or noeven by deleting k arguments, then the acceptance problems can be solved in time $3^k n^c$. The base 3 of the running time comes from the fact that the evaluation algorithm considers three different cases for the arguments in the backdoor set: (1) the argument is in the acceptable set, (2) the argument is not in the set and is attacked by at least one argument from the set, and (3) the argument is not in the set but is not attacked by any argument from the set.

12 Conclusion

Backdoor sets aim at exploiting hidden structures in real-world problem instances. The effectiveness of this approach has been investigated empirically in [31,42,63,88] and in many cases, small backdoor sets were found for large industrial instances.

As several backdoor set problems reduce to well-investigated core problems from parameterized complexity, such as VERTEX COVER, 3-HITTING SET, FEEDBACK VERTEX SET, and their variants, a few decades of focused research efforts can be used to detect backdoor sets efficiently. Nevertheless, several questions remain open. In particular, the parameterized complexity classification of several permissive problems seems challenging. As discussed at the end of Subsection 11.3, the classification of variants of the FEEDBACK VERTEX SET problem would also shed some light on backdoor set detection problems in nonmonotonic reasoning.

We believe that more research in this direction is necessary if we want to explain the good practical performance of heuristic SAT solvers. Directions for future research could involve multivariate parameterizations of backdoor problems and the consideration of backdoors to combinations of different base classes.

Acknowledgment. We thank Ryan Williams for his comments on an earlier version of this survey.

References

1. Abu-Khzam, F.N.: A kernelization algorithm for d-hitting set. *J. of Computer and System Sciences* 76(7), 524–531 (2010)
2. Aspvall, B., Plass, M.F., Tarjan, R.E.: A linear-time algorithm for testing the truth of certain quantified Boolean formulas. *Information Processing Letters* 8(3), 121–123 (1979)
3. Bacchus, F., Dalmao, S., Pitassi, T.: Algorithms and complexity results for #SAT and Bayesian inference. In: 44th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2003), pp. 340–351 (2003)
4. Baroni, P., Giacomin, M.: Semantics of abstract argument systems. In: Rahwan, I., Simari, G. (eds.) *Argumentation in Artificial Intelligence*, pp. 25–44. Springer (2009)
5. Bench-Capon, T.J.M., Dunne, P.E.: *Argumentation in artificial intelligence*. *Artificial Intelligence* 171(10-15), 619–641 (2007)
6. Berre, D.L., Parrain, A.: On SAT technologies for dependency management and beyond. In: Thiel, S., Pohl, K. (eds.) *Proceedings of 12th International Conference Software Product Lines Workshops, SPLC 2008, Limerick, Ireland, September 8-12, vol. 2*, pp. 197–200. Lero Int. Science Centre, University of Limerick, Ireland (2008)
7. Besnard, P., Hunter, A.: *Elements of Argumentation*. The MIT Press (2008)
8. Bjesse, P., Leonard, T., Mokkedem, A.: Finding Bugs in an Alpha Microprocessor Using Satisfiability Solvers. In: Berry, G., Comon, H., Finkel, A. (eds.) *CAV 2001. LNCS, vol. 2102*, pp. 454–464. Springer, Heidelberg (2001)
9. Bodlaender, H.L.: On disjoint cycles. *International Journal of Foundations of Computer Science* 5(1), 59–68 (1994)

10. Bodlaender, H.L., Downey, R.G., Fellows, M.R., Hermelin, D.: On problems without polynomial kernels. *J. of Computer and System Sciences* 75(8), 423–434 (2009)
11. Bondarenko, A., Dung, P.M., Kowalski, R.A., Toni, F.: An abstract, argumentation-theoretic approach to default reasoning. *Artificial Intelligence* 93(1-2), 63–101 (1997)
12. Bonsma, P., Lokshtanov, D.: Feedback Vertex Set in Mixed Graphs. In: Dehne, F., Iacono, J., Sack, J.-R. (eds.) WADS 2011. LNCS, vol. 6844, pp. 122–133. Springer, Heidelberg (2011)
13. Kleine Büning, H., Karpinski, M., Flögel, A.: Resolution for quantified Boolean formulas. *Information and Computation* 117(1), 12–18 (1995)
14. Cai, L., Huang, X.: Fixed-parameter approximation: Conceptual framework and approximability results. *Algorithmica* 57(2), 398–412 (2010)
15. Chavira, M., Darwiche, A.: On probabilistic inference by weighted model counting. *Artificial Intelligence* 172(6-7), 772–799 (2008)
16. Chen, J., Kanj, I.A.: On approximating minimum vertex cover for graphs with perfect matching. *Theoretical Computer Science* 337(1-3), 305–318 (2005)
17. Chen, J., Fomin, F.V., Liu, Y., Lu, S., Villanger, Y.: Improved algorithms for feedback vertex set problems. *J. of Computer and System Sciences* 74(7), 1188–1198 (2008)
18. Chen, J., Kanj, I.A., Jia, W.: Vertex cover: further observations and further improvements. *J. Algorithms* 41(2), 280–301 (2001)
19. Chen, J., Kanj, I.A., Xia, G.: Improved upper bounds for vertex cover. *Theoretical Computer Science* 411(40–42), 3736–3756 (2010)
20. Chen, J., Liu, Y., Lu, S., O’Sullivan, B., Razgon, I.: A fixed-parameter algorithm for the directed feedback vertex set problem. *J. of the ACM* 55(5), Art. 21, 19 (2008)
21. Chen, Y.-J., Grohe, M., Grüber, M.: On Parameterized Approximability. In: Bodlaender, H.L., Langston, M.A. (eds.) IWPEC 2006. LNCS, vol. 4169, pp. 109–120. Springer, Heidelberg (2006)
22. Cook, S.A.: The complexity of theorem-proving procedures. In: Proc. 3rd Annual Symp. on Theory of Computing, pp. 151–158. Shaker Heights, Ohio (1971)
23. Cook, S.A., Mitchell, D.G.: Finding hard instances of the satisfiability problem: a survey. In: Satisfiability problem: theory and applications, Piscataway, NJ. American Mathematical Society, pp. 1–17 (1997)
24. Coste-Marquis, S., Devred, C., Marquis, P.: Symmetric Argumentation Frameworks. In: Godo, L. (ed.) ECSQARU 2005. LNCS (LNAI), vol. 3571, pp. 317–328. Springer, Heidelberg (2005)
25. Courcelle, B., Makowsky, J.A., Rotics, U.: On the fixed parameter complexity of graph enumeration problems definable in monadic second-order logic. *Discr. Appl. Math.* 108(1-2), 23–52 (2001)
26. Courcelle, B.: Graph rewriting: an algebraic and logic approach. In: Handbook of Theoretical Computer Science, vol. B, pp. 193–242. Elsevier Science Publishers, North-Holland (1990)
27. Crama, Y., Ekin, O., Hammer, P.L.: Variable and term removal from Boolean formulae. *Discr. Appl. Math.* 75(3), 217–230 (1997)
28. Cygan, M., Pilipczuk, M., Pilipczuk, M., Wojtaszczyk, J.O.: Subset Feedback Vertex Set Is Fixed-Parameter Tractable. In: Aceto, L., Henzinger, M., Sgall, J. (eds.) ICALP 2011. LNCS, vol. 6755, pp. 449–461. Springer, Heidelberg (2011)
29. Davis, M., Putnam, H.: A computing procedure for quantification theory. *J. of the ACM* 7(3), 201–215 (1960)

30. Davis, M., Logemann, G., Loveland, D.: A machine program for theorem-proving. *Communications of the ACM* 5, 394–397 (1962)
31. Dilkina, B.N., Gomes, C.P., Sabharwal, A.: Tradeoffs in the Complexity of Backdoor Detection. In: Bessière, C. (ed.) *CP 2007*. LNCS, vol. 4741, pp. 256–270. Springer, Heidelberg (2007)
32. Dimopoulos, Y., Torres, A.: Graph theoretical structures in logic programs and default theories. *Theoretical Computer Science* 170(1-2), 209–244 (1996)
33. Downey, R.G., Fellows, M.R.: *Parameterized Complexity*. Monographs in Computer Science. Springer, New York (1999)
34. Downey, R.G., Fellows, M.R., McCartin, C.: Parameterized Approximation Problems. In: Bodlaender, H.L., Langston, M.A. (eds.) *IWPEC 2006*. LNCS, vol. 4169, pp. 121–129. Springer, Heidelberg (2006)
35. Dung, P.M.: On the acceptability of arguments and its fundamental role in non-monotonic reasoning, logic programming and n -person games. *Artificial Intelligence* 77(2), 321–357 (1995)
36. Dunne, P.E.: Computational properties of argument systems satisfying graph-theoretic constraints. *Artificial Intelligence* 171(10-15), 701–729 (2007)
37. Dunne, P.E., Bench-Capon, T.J.M.: Coherence in finite argument systems. *Artificial Intelligence* 141(1-2), 187–203 (2002)
38. Eiter, T., Gottlob, G.: On the computational cost of disjunctive logic programming: propositional case. *Ann. Math. Artif. Intell.* 15(3-4), 289–323 (1995)
39. Fellows, M.R., Szeider, S., Wrightson, G.: On finding short resolution refutations and small unsatisfiable subsets. *Theoretical Computer Science* 351(3), 351–359 (2006)
40. Fernau, H.: A top-down approach to search-trees: Improved algorithmics for 3-hitting set. *Algorithmica* 57(1), 97–118 (2010)
41. Fichte, J.K.: The good, the bad, and the odd: Cycles in answer-set programs. In: *ESSLI 2011* (2011)
42. Fichte, J.K., Szeider, S.: Backdoors to tractable answer-set programming. Technical Report 1104.2788, Arxiv.org (2012), Extended and updated version of a paper that appeared in the proceedings of IJCAI 2011. The 22nd International Joint Conference on Artificial Intelligence (2012)
43. Fischer, E., Makowsky, J.A., Ravve, E.R.: Counting truth assignments of formulas of bounded tree-width or clique-width. *Discr. Appl. Math.* 156(4), 511–529 (2008)
44. Fortnow, L., Santhanam, R.: Infeasibility of instance compression and succinct PCPs for NP. In: Dwork, C. (ed.) *Proceedings of the 40th Annual ACM Symposium on Theory of Computing*, Victoria, British Columbia, Canada, May 17-20, pp. 133–142. ACM (2008)
45. Garey, M.R., Johnson, D.R.: *Computers and Intractability*. W. H. Freeman and Company, New York (1979)
46. Gaspers, S., Szeider, S.: Backdoors to acyclic SAT. In: *Proceedings of ICALP 2012 (Track A: Algorithms, Complexity and Games)*, the 39th International Colloquium on Automata, Languages and Programming, University of Warwick, UK, July 9-13. LNCS. Springer (to appear, 2012)
47. Gaspers, S., Szeider, S.: Strong backdoors to nested satisfiability. In: *Proceedings of SAT 2012, the 15th International Conference on Theory and Applications of Satisfiability Testing*, Trento, Italy, June 17-20, 2012. LNCS. Springer (to appear, 2012)
48. Gaspers, S., Szeider, S.: Strong backdoors to bounded treewidth SAT. Technical report 1204.6233, Arxiv.org (2012)

49. Gelfond, M., Lifschitz, V.: Classical negation in logic programs and disjunctive databases. *New Generation Comput.* 9(3/4), 365–386 (1991)
50. Gomes, C.P., Kautz, H., Sabharwal, A., Selman, B.: Satisfiability solvers. In: *Handbook of Knowledge Representation. Foundations of Artificial Intelligence*, vol. 3, pp. 89–134. Elsevier (2008)
51. Gottlob, G., Szeider, S.: Fixed-parameter algorithms for artificial intelligence, constraint satisfaction, and database problems. *The Computer Journal* 51(3), 303–325 (2006); survey paper
52. Guo, J., Hüffner, F., Niedermeier, R.: A Structural View on Parameterizing Problems: Distance from Triviality. In: Downey, R., Fellows, M., Dehne, F. (eds.) *IWPEC 2004. LNCS*, vol. 3162, pp. 162–173. Springer, Heidelberg (2004)
53. Hertli, T.: 3-SAT faster and simpler - unique-SAT bounds for PPSZ hold in general. In: Ostrovsky, R. (ed.) *Proceedings of the 52nd Annual IEEE Symposium on Foundations of Computer Science (FOCS 2011)*. IEEE (2011)
54. Iwama, K.: CNF-satisfiability test by counting and polynomial average time. *SIAM J. Comput.* 18(2), 385–391 (1989)
55. Kakimura, N., Kawarabayashi, K., Kobayashi, Y.: Erdős-Pósa property and its algorithmic applications: parity constraints, subset feedback set, and subset packing. In: Rabani, Y. (ed.) *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19*, pp. 1726–1736. SIAM (2012)
56. Kautz, H.A., Selman, B.: Planning as satisfiability. In: *Proceedings ECAI 1992*, pp. 359–363 (1992)
57. Kawarabayashi, K., Kobayashi, Y.: Fixed-parameter tractability for the subset feedback set problem and the s -cycle packing problem. Technical report, University of Tokyo, Japan (2010); see also [55]
58. Büning, H.K., Lettman, T.: *Propositional logic: deduction and algorithms*. Cambridge University Press, Cambridge (1999)
59. Knuth, D.E.: Nested satisfiability. *Acta Informatica* 28(1), 1–6 (1990)
60. Kolaitis, P.G., Vardi, M.Y.: Conjunctive-query containment and constraint satisfaction. *J. of Computer and System Sciences* 61(2), 302–332 (2000); Special issue on the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, Seattle, WA (1998)
61. Levin, L.: Universal sequential search problems. *Problems of Information Transmission* 9(3), 265–266 (1973)
62. Lewis, H.R.: Renaming a set of clauses as a Horn set. *J. of the ACM* 25(1), 134–135 (1978)
63. Li, Z., van Beek, P.: Finding Small Backdoors in SAT Instances. In: Butz, C., Lingras, P. (eds.) *Canadian AI 2011. LNCS*, vol. 6657, pp. 269–280. Springer, Heidelberg (2011)
64. Gupta, A., Prasad, M., Biere, A.: A survey of recent advances in SAT-based formal verification. *Software Tools for Technology Transfer* 7(2), 156–173 (2005)
65. Marek, V.W., Truszczynski, M.: Stable models and an alternative logic programming paradigm. In: *The Logic Programming Paradigm: a 25-Year Perspective*, pp. 169–181. Springer (1999)
66. Marx, D., Schlotter, I.: Parameterized complexity and local search approaches for the stable marriage problem with ties. *Algorithmica* 58(1), 170–187 (2010)
67. Marx, D., Schlotter, I.: Stable assignment with couples: parameterized complexity and local search. *Discrete Optim.* 8(1), 25–40 (2011)

68. Mishra, S., Raman, V., Saurabh, S., Sikdar, S., Subramanian, C.R.: The Complexity of Finding Subgraphs Whose Matching Number Equals the Vertex Cover Number. In: Tokuyama, T. (ed.) ISAAC 2007. LNCS, vol. 4835, pp. 268–279. Springer, Heidelberg (2007)
69. Niemelä, I.: Logic programs with stable model semantics as a constraint programming paradigm. *Ann. Math. Artif. Intell.* 25(3-4), 241–273 (1999); Logic programming with non-monotonic semantics: representing knowledge and its computation
70. Nishimura, N., Ragde, P., Szeider, S.: Detecting backdoor sets with respect to Horn and binary clauses. In: Proceedings of SAT 2004 Seventh International Conference on Theory and Applications of Satisfiability Testing, Vancouver, BC, Canada, May 10-13, pp. 96–103 (2004)
71. Nishimura, N., Ragde, P., Szeider, S.: Solving #SAT using vertex covers. *Acta Informatica* 44(7-8), 509–523 (2007)
72. Ordyniak, S., Paulusma, D., Szeider, S.: Satisfiability of acyclic and almost acyclic CNF formulas. In: Lodaya, K., Mahajan, M. (eds.) IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2010, Chennai, India, December 15-18. LIPIcs, vol. 8, pp. 84–95. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2010)
73. Ordyniak, S., Szeider, S.: Augmenting tractable fragments of abstract argumentation. In: Walsh, T. (ed.) Proceedings of the 22nd International Joint Conference on Artificial Intelligence, IJCAI 2011, pp. 1033–1038. AAAI Press (2011)
74. Otwell, C., Remshagen, A., Truemper, K.: An effective QBF solver for planning problems. In: Proceedings of MSV/AMCS, pp. 311–316. CSREA Press (2004)
75. Pan, G., Vardi, M.Y.: Fixed-parameter hierarchies inside PSPACE. In: Proceedings of 21th IEEE Symposium on Logic in Computer Science (LICS 2006), Seattle, WA, USA, August 12-15, pp. 27–36. IEEE Computer Society Press (2006)
76. Papadimitriou, C.H.: Computational Complexity. Addison-Wesley (1994)
77. Parsons, S., Wooldridge, M., Amgoud, L.: Properties and complexity of some formal inter-agent dialogues. *J. Logic Comput.* 13(3), 347–376 (2003)
78. Pietrzak, K.: On the parameterized complexity of the fixed alphabet shortest common supersequence and longest common subsequence problems. *J. of Computer and System Sciences* 67(4), 757–771 (2003)
79. Pollock, J.L.: How to reason defeasibly. *Artificial Intelligence* 57(1), 1–42 (1992)
80. Rahwan, I., Simari, G.R. (eds.): Argumentation in Artificial Intelligence. Springer (2009)
81. Raman, V., Saurabh, S., Subramanian, C.R.: Faster fixed parameter tractable algorithms for finding feedback vertex sets. *ACM Transactions on Algorithms* 2(3), 403–415 (2006)
82. Razgon, I., O’Sullivan, B.: Almost 2-SAT is fixed parameter tractable. *J. of Computer and System Sciences* 75(8), 435–450 (2009)
83. Reed, B., Smith, K., Vetta, A.: Finding odd cycle transversals. *Oper. Res. Lett.* 32(4), 299–301 (2004)
84. Rintanen, J.: Constructing conditional plans by a theorem-prover. *J. Artif. Intell. Res.* 10, 323–352 (1999)
85. Robertson, N., Seymour, P.D., Thomas, R.: Permanents, Pfaffian orientations, and even directed circuits. *Ann. of Math (2)* 150(3), 929–975 (1999)
86. Roth, D.: On the hardness of approximate reasoning. *Artificial Intelligence* 82(1-2), 273–302 (1996)

87. Sabharwal, A., Ansotegui, C., Gomes, C.P., Hart, J.W., Selman, B.: QBF Modeling: Exploiting Player Symmetry for Simplicity and Efficiency. In: Biere, A., Gomes, C.P. (eds.) SAT 2006. LNCS, vol. 4121, pp. 382–395. Springer, Heidelberg (2006)
88. Samer, M., Szeider, S.: Backdoor trees. In: Twenty-Third Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, July 13–17, pp. 363–368. AAAI Press (2008)
89. Samer, M., Szeider, S.: Backdoor sets of quantified Boolean formulas. *Journal of Automated Reasoning* 42(1), 77–97 (2009)
90. Samer, M., Szeider, S.: Fixed-parameter tractability. In: Biere, A., Heule, M., van Maaren, H., Walsh, T. (eds.) *Handbook of Satisfiability*, vol. ch. 13, pp. 425–454. IOS Press (2009)
91. Samer, M., Szeider, S.: Algorithms for propositional model counting. *J. Discrete Algorithms* 8(1), 50–64 (2010)
92. Sang, T., Beame, P., Kautz, H.A.: Performing bayesian inference by weighted model counting. In: *Proceedings, The Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference*, Pittsburgh, Pennsylvania, USA, July 9–13, pp. 475–482. AAAI Press / The MIT Press (2005)
93. Schaefer, T.J.: The complexity of satisfiability problems. In: *Conference Record of the Tenth Annual ACM Symposium on Theory of Computing*, San Diego, Calif., pp. 216–226. ACM (1978)
94. Stockmeyer, L.J., Meyer, A.R.: Word problems requiring exponential time. In: *Proc. Theory of Computing*, pp. 1–9. ACM (1973)
95. Szeider, S.: On Fixed-Parameter Tractable Parameterizations of SAT. In: Giunchiglia, E., Tacchella, A. (eds.) SAT 2003. LNCS, vol. 2919, pp. 188–202. Springer, Heidelberg (2004)
96. Szeider, S.: Backdoor sets for DLL subsolvers. *Journal of Automated Reasoning* 35(1-3), 73–88 (2005); Reprinted as Giunchiglia, E., Walsh, T.(eds.): SAT 2005 - Satisfiability Research in the Year 2005, ch. 4. Springer Verlag (2006)
97. Szeider, S.: Matched formulas and backdoor sets. *J. on Satisfiability, Boolean Modeling and Computation* 6, 1–12 (2008)
98. Szeider, S.: Limits of preprocessing. In: *Proceedings of the Twenty-Fifth Conference on Artificial Intelligence, AAAI 2011*, pp. 93–98. AAAI Press, Menlo Park (2011)
99. Velev, M.N., Bryant, R.E.: Effective use of Boolean satisfiability procedures in the formal verification of superscalar and VLIW microprocessors. *J. Symbolic Comput.* 35(2), 73–106 (2003)
100. Weld, D.S.: Recent advances in AI planning. *AI Magazine* 20(2), 93–123 (1999)
101. Williams, R., Gomes, C., Selman, B.: Backdoors to typical case complexity. In: Gottlob, G., Walsh, T. (eds.) *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, IJCAI 2003*, pp. 1173–1178. Morgan Kaufmann (2003)

Studies in Computational Aspects of Voting

A Parameterized Complexity Perspective*

Dedicated to Michael R. Fellows on the occasion of his 60th birthday

Nadja Betzler, Robert Brederick, Jiehua Chen, and Rolf Niedermeier

Institut für Softwaretechnik und Theoretische Informatik,
TU Berlin, Germany
{robert.brederick,jiehua.chen,rolf.niedermeier}@tu-berlin.de

Abstract. We review NP-hard voting problems together with their status in terms of parameterized complexity results. In addition, we survey standard techniques for achieving fixed-parameter (in)tractability results in voting.

1 Introduction

Once there is more than one alternative for a community to choose from, voting comes into play. Different voters usually have conflicting preferences over the alternatives, hence some voting protocol has to be used to reach a joint decision or, in other words, to aggregate preferences. Voting is part of the fields of preference handling, decision making, and social choice. There are many voting protocols whose pros and cons have been studied for centuries in such diverse fields as philosophy, mathematics, political science, and economy. Recently, computer science has entered the stage for several reasons. With the omnipresence of the Internet and modern communication tools, applications such as auctions, bids, ratings, and rankings have become an everyday business. All these are related to voting scenarios. Moreover, the advent of intelligent multi-agent systems leads to numerous cases of preference aggregation. Inside computer science, voting occurs in quite diverse areas, including planning problems in multi-agent systems [ER91, ER97], spam detection [DKNS01a], databases [FKS03], bioinformatics [JSA08], and graph drawing [BBD09]. We refer interested readers to a couple of surveys [BCE12, BEH⁺10, CELM07, Con10, FHH10, FHHR09a] and a book [RBLR11, in German] for a general overview on voting in computer science.

Voting problems (winner determination being just the most basic one) come in many different guises, often making the corresponding tasks computationally challenging to solve. First of all, there are numerous different voting protocols including Plurality, k -Approval, and Kemeny, to name just a few. Then, it may happen that there are only incomplete voter preferences available, making the determination of a possible or necessary winner hard. Moreover, questions such as manipulation, control, or bribery often lead to NP-hard problems. The study

* Supported by the DFG, research project PAWS, NI 369/10.

of the computational complexity of voting problems was initiated by a seminal series of papers of Bartholdi, Orlin, Tovey, and Trick [BO91, BTT89a, BTT89b]. Many voting problems turned out to be NP-hard. Actually, Bartholdi et al. pointed out that in the context of voting, computational intractability may sometimes be a desirable property. For instance, it is desirable to have a voting protocol that is “resistant” against attacks such as manipulation or bribery.

Voting problems carry many natural parameters, obviously including the number of candidates and the number of votes. There are real-world scenarios for each of them having small values. Hence, the analysis of parameterized computational complexity comes into play. To the best of our knowledge, this fruitful line of research was *explicitly* initiated Christian, Fellows, Rosamond, and Slinko [CFRS07] in a work concerned with lobbying. Moreover, a complexity analysis for manipulating voting systems when the parameter “number of candidates” is small was addressed by Conitzer, Sandholm, and Lang [CSL07]. In this survey, we try to review the state of the art and motivate the rapidly developing field of parameterized complexity analysis for voting problems. See Lindner and Rothe [LR08] for an early survey in this direction.

Our work is organized as follows. Section 2 introduces some basic concepts and definitions related to both voting problems and parameterized complexity analysis. In Section 3, we briefly review a number of prominent voting protocols and some of their respective pros and cons. In Section 4, we survey in some detail the state of the art concerning the multivariate complexity analysis for Kemeny voting. This exhibits how many different parameters naturally occur in a practically relevant voting problem, and how the tools of parameterized complexity analysis can help to better understand the computational complexity of an NP-hard voting problem. In Section 5, we present several NP-hard voting problems and describe their status in terms of parameterized complexity analysis. In Section 6, we describe applications of tools from parameterized algorithmics that have been applied to gain fixed-parameter tractability results for voting problems. Finally, in Section 7, we discuss the relevance and benefits of parameterized (and multivariate) complexity analysis in voting scenarios and conclude with numerous challenges for future research.

2 Preliminaries

Since we are talking about voting problems and their computational complexity, we start with basic definitions from the context of voting. We assume familiarity with *classical computational complexity theory* [Pap94, AB09], and we provide some basic definitions concerning *parameterized computational complexity theory* [DF99, FG06, Nie06].

Formally, an *election* (C, V) consists of a set C of m candidates (or, synonymously, alternatives) and a multiset V of n votes. If not stated otherwise, a *vote* is a linear order (that is, a transitive, antisymmetric, and total relation) on C . Sometimes we also call this a *ranking* over C . For example, for $C = \{a, b, c\}$, the vote $a \succ_v b \succ_v c$ expresses that a is the best-liked and c the least-liked candidate

in the vote v . We use ‘ \succ ’ instead of ‘ \succ_v ’ if it is clear from the context which vote we mean. For any two candidates $a \neq b$, let $\#(a, b)$ be the number of votes that rank candidate a higher than candidate b in the considered election. A *voting protocol*¹ is a function that maps an election to a subset of candidates, the set of winners. When one is interested in finding a uniquely determined winner (that is, a one-element winner set), one refers to such a candidate as *unique winner*. When allowing for a set of winners, the corresponding candidates are denoted as *co-winners*.

Sometimes we also consider a more general definition of votes. There are scenarios where (complete) linear orders are not available. That is, some candidates are not comparable in some votes, leading to *incomplete votes*. In such cases, our votes are partial orders on the candidate set. A linear order v *extends* a partial order w if $w \subseteq v$, that is, for any $c_1, c_2 \in C$ one has $c_1 \succ_w c_2 \Rightarrow c_1 \succ_v c_2$. Consider two candidates $a, b \in C$ and an incomplete vote $v \in V$. If neither $a \succ_v b$ nor $b \succ_v a$, then we say the candidate pair $\{a, b\}$ is *undetermined* in vote v .

Parameterized Complexity. The concept of parameterized complexity was pioneered by Downey and Fellows [DF99] (see also [FG06, Nie06] for more recent textbooks). The fundamental goal is to find out whether the seemingly unavoidable combinatorial explosion occurring in algorithms to decide NP-hard problems can be confined to certain problem-specific parameters. The idea is the following: When such a parameter assumes only small values in applications, then an algorithm with a running time that is exponential exclusively with respect to the parameter may be efficient and practical. We now provide some formal definitions.

Definition 1 (Parameterized Problem). *A parameterized problem is a language $L \subseteq \Sigma^* \times \Sigma^*$, where Σ is an alphabet. The second component is called the parameter of the problem.*

We typically consider the special case of parameters which are non-negative integers or “combined” parameters which are tuples of non-negative integers. For instance, an obvious parameter in voting is the number of candidates. Thus, typically $L \subseteq \Sigma^* \times \mathbb{N}$, where a combined parameter can be interpreted as the maximum of its integer components.

Definition 2 (Fixed-Parameter Tractability). *A parameterized problem L is fixed-parameter tractable if there is an algorithm that decides in $f(k) \cdot |x|^{\mathcal{O}(1)}$ time whether $(x, k) \in L$, where f is an arbitrary computable function depending only on k . Correspondingly, FPT denotes the class of all fixed-parameter tractable parameterized problems.*

¹ In this survey, we do not discuss the more general concepts of *social choice functions* or *social welfare functions*. Note that by our definition of voting protocols, every voting protocol is *anonymous*, that is, the voting protocol does not discriminate among voters. We will only exemplarily discuss some other properties of the considered voting protocols when necessary. For an overview about general concepts and properties of voting protocols, we refer to the two handbooks on social choice and welfare [ASS02, ASS10].

We stress that the concept of fixed-parameter tractability is different from the notion of “polynomial-time solvability for constant k ” since an algorithm running in $\mathcal{O}(|x|^k)$ time does not show fixed-parameter tractability. All problems which can be solved in running time $|x|^{f(k)}$ for a computable function f form the complexity class XP, where $f : \mathbb{N} \rightarrow \mathbb{N}$ is a function depending only on k . Clearly, $\text{FPT} \subseteq \text{XP}$.

For many parameterized problems, fixed-parameter tractability could not be shown. Downey and Fellows [DF99] developed a theory of (presumable) *parameterized intractability*. It comprises of a hierarchy of complexity classes coming along with complete problems. This so-called *W-hierarchy* consists of the following classes and interrelations:

$$\text{FPT} \subseteq \text{W}[1] \subseteq \text{W}[2] \subseteq \dots \subseteq \text{W}[\text{Sat}] \subseteq \text{W}[\text{P}] \subseteq \text{XP}.$$

In this survey, we only provide intractability results regarding the first two levels of (presumable) parameterized intractability captured by the complexity classes $\text{W}[1]$ and $\text{W}[2]$. The containment $\text{W}[1] \subseteq \text{FPT}$ would not imply $\text{P} = \text{NP}$ but the failure of the Exponential Time Hypothesis [IP01, IPZ01].² It is commonly believed that $\text{W}[1]$ -hard problems are not fixed-parameter tractable. To show $\text{W}[t]$ -hardness for any non-negative integer t , we introduce the following reducibility concept.

Definition 3 (Parameterized Reduction). *Let $L, L' \subseteq \Sigma^* \times \mathbb{N}$ be two parameterized problems. A parameterized reduction from L to L' consists of two mappings $\phi : \Sigma^* \times \mathbb{N} \rightarrow \Sigma^*$ and $g : \mathbb{N} \rightarrow \mathbb{N}$, where for every $x \in \Sigma^*$ and $k \in \mathbb{N}$ it holds that*

- $(x, k) \mapsto \phi(x, k)$ is computable in time $h(k) \cdot |x|^{\mathcal{O}(1)}$ with $h : \mathbb{N} \rightarrow \mathbb{N}$, and
- $(x, k) \in L \Leftrightarrow (\phi(x, k), g(k)) \in L'$.

Analogously to the case of NP-hardness, for any non-negative integer t , it suffices to give a parameterized reduction from one $\text{W}[t]$ -hard parameterized problem X to a parameterized problem Y to show $\text{W}[t]$ -hardness of Y . Containment of Y in $\text{W}[t]$ can be shown by giving a reduction from Y to a problem contained in $\text{W}[t]$. If there are parameterized reductions for two problems such that each of them can be reduced to the other problem, we say that they are *FPT-equivalent*.

Kernelization [Bod09, GN07] is an alternative way of showing fixed-parameter tractability [CCDF97]. In a nutshell, it is a polynomial-time algorithm that transforms an instance of a parameterized problem into an equivalent instance whose size is bounded by a function of the parameter. This resulting instance is called a *problem kernel*. Typically, kernelizations are based on several polynomial-time executable *data-reduction rules* that help shrinking the instance size.

For more details about parameterized complexity theory we refer to the textbooks [DF99, FG06, Nie06] and a recent survey by Downey and Thilikos [DT11].

² In a nutshell, the Exponential Time Hypothesis says that, for $k \geq 3$, the NP-complete k -SAT problem cannot be solved in time subexponential in the number of variables.

Table 1. Hypothetical student rankings of TU Berlin (B), MIT (M), Oxford University (O), Tsinghua University (T) and ETH Zurich (Z) according to research in parameterized complexity, salary, practicing English, and cultural activities, respectively

Criterion	Institutions
Parameterized complexity	$B \succ O \succ M \succ T \succ Z$
Salary	$Z \succ O \succ M \succ T \succ B$
Practicing English	$M \succ O \succ B \succ Z \succ T$
Cultural activities	$B \succ T \succ Z \succ M \succ O$

3 Types of Voting Protocols

Suppose that a student decides to pursue his PhD in parameterized complexity analysis. He gets five offers: From TU Berlin (B), MIT (M), Oxford University (O), Tsinghua University (T), and ETH Zurich (Z). He decides to select one that is not only good in research but also offers a manifold of cultural activities, as well as a good opportunity to polish his English. Last but not least, he needs a decent income.

Depending on these different criteria, the five universities are ranked³. In the field of parameterized complexity, TU Berlin is ranked first, followed by Oxford University, MIT, and Tsinghua University. ETH Zurich is ranked last. As for salaries, ETH Zurich makes the best offer, followed by Oxford University, MIT, and Tsinghua University. TU Berlin offers the least. For practicing English, MIT ranks first, followed by Oxford University, and then by TU Berlin. ETH Zurich is ranked fourth and Tsinghua University last. With respect to cultural activities, TU Berlin is ranked the best, followed by Tsinghua University, and then by ETH Zurich. Oxford University is ranked last, just behind MIT. These rankings are listed in Table 1.

The universities B, M, O, T, and Z can be seen as the candidates for an election and the rankings in Table 1 as the votes on these candidates. Deciding on an institution means aggregating the different rankings and deciding on the winner of this election.

Applying different voting protocols to the same multiset of votes may lead to different winners. Many of the most widely used voting protocols can be assigned to one of the following two classes: *scoring protocols* and *voting protocols based on pairwise comparisons between candidates*. In the following, we will look into these two classes in some detail (Sections 3.1 and 3.2). In Section 3.3, we will take a look at some additional voting protocols which fall into neither class. We illustrate some common and popular voting protocols with the help of our PhD place example. We emphasize that it would be beyond the scope of our survey to name

³ Clearly, these rankings are influenced by marketing and political pressure. In this example, also a certain degree of bribery comes into play.

and discuss the properties (desirable and undesirable ones) of the various voting protocols. For further information on this topic, or voting protocols in general, we refer to the expositions of Arrow et al. [ASS02, ASS10], Gaertner [Gae09], Nurmi [Nur87], Rothe et al. [RBLR11, in German], and Taylor [Tay05].

3.1 Scoring Protocols

In a (positional) scoring protocol, each candidate is assigned a certain number of points from each vote depending on her position in this vote. A candidate is called a winner if no other candidate gets a higher total sum of points.

Plurality. Plurality is perhaps the most widely used voting protocol. It is a scoring protocol which assigns one point to the top-ranked candidate in each vote, and zero points to all others. A candidate with the highest total score belongs to the winner set. There may be multiple winners. In our PhD place example, it is easily seen that the winning university is TU Berlin (2 points) if we use the Plurality protocol to make the decision.

Plurality is very simple. However, it has some shortcomings. For example, it is often criticized for considering only the topmost candidate of each vote and completely disregarding the information about other candidates. For this reason voters sometimes do not submit their true preferences if they know that their most preferred candidate has only a small chance to win. Suppose that there is an election on three candidates, a , b , and c , with

two votes $a \succ b \succ c$,
 four votes $c \succ b \succ a$, and
 three vote $b \succ a \succ c$.

According to the Plurality voting protocol, candidate c wins with four points. However, if the first two voters exchange the positions of candidates a and b in their votes such that they submit $b \succ a \succ c$ instead of $a \succ b \succ c$, then candidate b wins with five points. This is a better outcome for them, since they prefer candidate b to candidate c .

k-Approval. Occasionally, a voter has more than one favorite candidate. The k -Approval voting protocol gives the possibility to “approve” k candidates: The first k candidates in a vote get one point each. Thus, Plurality is the same as 1-Approval. In our example, using 2-Approval, one would select Oxford University (3 points) for a PhD position, which intuitively seems to be a good compromise, since three of four criteria are ranking Oxford University in the second position.

Veto. Another simple scoring protocol is Veto. It assigns zero points to the last candidate and one point to each of the other candidates in each vote. Once again, every candidate with the highest sum of points wins. Using Veto, the PhD place example will result in selecting MIT (4 points). Veto is the same as $(m - 1)$ -Approval, where m is the total number of candidates.

Borda. A prominent voting protocol is Borda voting.⁴ Borda voting directly translates the position of each candidate in a vote into the number of points she gets. For each vote, Borda voting assigns zero points to the candidate ranked last, one point to the candidate ranked last but one, etc., and the highest-ranked candidate in each vote is assigned $m - 1$ points. Once again, every candidate with the highest total score wins. According to Borda voting, TU Berlin (10 points) is the winner in the PhD place example.

Determining the set of winners using any of the scoring protocols described above can be easily done in time polynomial in the input size.

3.2 Voting Protocols Based on Pairwise Comparisons

Comparison-based voting protocols date back to the 13th century. Ramon Lull, who first came up with Borda voting, devised a voting protocol which takes into account pairwise comparisons between any two candidates. Today, this is known as the *Condorcet method* [dC85].⁵

Definition 4 (Condorcet Winner). *A candidate is the Condorcet winner if she is preferred to any other candidate in more than half of the votes.*

Obviously, deciding whether a candidate is the Condorcet winner can be done efficiently, that is, in polynomial time. However, not every election has a Condorcet winner. For instance, in the PhD place example, there is no Condorcet winner since no institution has an absolute majority of votes which prefer it to any other institution: TU Berlin beats both, Tsinghua University and ETH Zurich by 3-to-1; TU Berlin and Oxford University, TU Berlin and MIT as well as Oxford University and MIT are tied 2-to-2. The pairwise comparisons of every two candidates are shown in Table 2.

There is a close relation between directed graphs and voting protocols based on pairwise comparisons. More precisely, for each election there is a *majority graph* which is defined as follows:

Definition 5 (Majority Graph). *The majority graph of an election $E = (C, V)$ is a directed graph whose vertices are the candidates and there is an arc from vertex v to vertex w if and only if more than half of the votes prefer candidate v to candidate w . An arc from v to w is labeled with “ $x : y$ ” which means that x votes prefer v to w , and y votes prefer w to v .*

Many voting problems (especially when comparison-based voting protocols are involved) can be considered as directed (weighted) graph problems. For example, if there is a vertex with exactly $m - 1$ outgoing arcs with m being the number

⁴ The Borda voting protocol was invented independently several times. It was first described by Ramon Lull, a 13th century Majorcan writer and philosopher. It is now named after Jean-Charles de Borda, a French mathematician, physicist, political scientist of the 18th century [dB81].

⁵ Named after the 18th-century French philosopher Marie Jean Antonie Nicolas de Caritat, Marquis de Condorcet.

Table 2. Pairwise comparisons in the PhD place example

Candidate pairs (x, y)	# votes with $x \succ y$	# votes with $y \succ x$
(B, O)	2	2
(B, M)	2	2
(B, T)	3	1
(B, Z)	3	1
(O, M)	2	2
(O, T)	3	1
(O, Z)	2	2
(M, T)	3	1
(M, Z)	2	2
(T, Z)	2	2

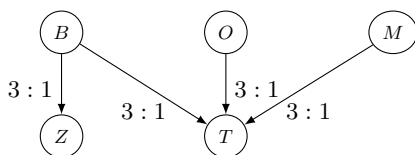


Fig. 1. The majority graph of our PhD place example

of candidates, then the corresponding candidate is the Condorcet winner. As we can easily see from the majority graph of our example in Figure 1, no vertex with out-degree four exists. This meets with the fact that there is no Condorcet winner in our PhD place example.

Although the existence of a Condorcet winner cannot be guaranteed, the Condorcet winner for an election is always unique if it does exist. Many voting protocols are designed to choose a candidate as the winner who is “closest” to the Condorcet winner. In the following, we will take a closer look at five well-known comparison-based voting protocols (Dodgson, Kemeny, Young, Copeland^α, and Maximin [Dod76, Kem59, You77, Cop51, Wal49, Fis77]) which all fulfill the *Condorcet principle*, that is, the Condorcet winner for an election will be selected as the winner if she exists. The winner determination problems for the first three voting protocols are NP-hard, while the last two can be solved efficiently, that is, in time polynomial in the input size.

Dodgson Voting. In his work “A Method of Taking Votes on More than Two Issues” [Dod76], the English writer, mathematician, and logician Charles Lutwidge Dodgson (better known as Lewis Carroll) proposed selecting the winner set as follows: Any candidate requiring the minimum number of *swaps* between two

neighboring candidates to become a Condorcet winner is considered as a winner. Given an election and a non-negative integer $k \in \mathbb{N}$, determining whether a candidate can become a Condorcet winner with at most k swaps in the given votes is NP-complete [BTT89b]. This problem is called DODGSON SCORE. In the PhD place example, the Dodgson score of TU Berlin is 2: By exchanging the positions of TU Berlin and Oxford University, and then the positions of TU Berlin and MIT, the ranking with respect to “Practicing English” turns into $B \succ M \succ O \succ Z \succ T$ and TU Berlin becomes the Condorcet winner. In fact, this is the fewest number of swaps needed to let the PhD place example have a Condorcet winner. Finally, we remark that generalized winner determination for Dodgson is complete for parallel access to NP ($\mathbf{P}_{\parallel}^{\text{NP}}$ -complete) [HHR97].

Kemeny Voting. This voting protocol goes back to Kemeny [Kem59] and Condorcet [dC85] and was specified by Levensick [Lev75] (see also our case study for Kemeny voting in Section 4). Consider an election consisting of a multiset of rankings of the candidates. The *Kendall-Tau distance* between rankings r_1 and r_2 is the number of swaps of two neighboring candidates in order to transform r_1 into r_2 . The *score* of a ranking r is the sum of Kendall-Tau distances between r and each input ranking. A “consensus ranking” with respect to Kemeny voting is a ranking with minimum score. Correspondingly, we call such a ranking a *Kemeny consensus*. The first candidate in a Kemeny consensus is considered as a winner. TU Berlin as well as Oxford University are winners in our PhD place example. See Section 4 for more details.

Kemeny voting has many desirable properties. For example, it is the only voting protocol which is *neutral* and *consistent*⁶, and satisfies the Condorcet principle [YL78]. Thus, Kemeny voting is used in various applications such as meta-search engines, spam detection [DKNS01a], databases [FKS03, Scu07], or the construction of genetic maps in bioinformatics [JSA08]. However, to determine a Kemeny consensus is computationally intractable. More precisely, the KEMENY SCORE problem, that is, given an election and a non-negative integer $k \in \mathbb{N}$, determining whether the score of a Kemeny consensus is at most k is NP-complete [BTT89b]. Some more general Kemeny voting related problems (including winner determination) are even $\mathbf{P}_{\parallel}^{\text{NP}}$ -complete [HSV05].

Young Voting. H. Peyton Young [You77] took a different approach to finding a candidate “closest” to the Condorcet winner. His main idea was to delete the fewest number of votes to let the remaining votes have a Condorcet winner. The YOUNG SCORE problem asks whether a candidate can become the Condorcet winner in a “sub-election” consisting of at least k' ($k' \in \mathbb{N}$) of the input votes,

⁶ A voting protocol is *neutral* if the candidates are treated equally, that is, if the candidates of an election are renamed, the winner of the election with renamed candidates is the renamed winner of the original election. *Consistency* requires that if a candidate wins in two multisets of votes, then she should also win in the union multiset of these two multisets.

while the DUAL YOUNG SCORE problem asks whether deleting at most k ($k \in \mathbb{N}$) votes can make a distinguished candidate the Condorcet winner. Both problems are NP-complete [RSV03]. For the PhD place example, removing only one vote (the ranking for practicing English) can make the remaining votes have a Condorcet winner (TU Berlin). Finally, we remark that the YOUNG WINNER problem, that is, deciding whether a distinguished candidate can become a Condorcet winner by the deleting minimum number of votes, is $\mathbf{P}_{\parallel}^{\mathbf{NP}}$ -complete [RSV03]. As one can easily verify, in our PhD place example, TU Berlin is a Young winner.

The NP-hardness results for the winner determination problems described above make such voting protocols usually infeasible for practical use. However, in some restricted scenarios winner determination becomes efficiently solvable. For example, DODGSON SCORE is fixed-parameter tractable with respect to the number of candidates. Table 4 in Section 5.1 gives some parameterized complexity results for DODGSON SCORE, DUAL YOUNG SCORE, and YOUNG SCORE, while the corresponding analysis of Kemeny voting is discussed in more detail in our case study (Section 4).

Copeland $^{\alpha}$ Voting. This voting protocol considers each pair of candidates: The candidate that beats the other one in more than half of the votes is rewarded one point, while the loser gets zero points. If the two candidates are tied, then each gets α points. The candidate with highest total score wins. The original Copeland voting [Cop51, BF02, Goo54] uses a slightly different way for awarding points to the loser in a pairwise comparison. However, it is equivalent to Copeland $^{0.5}$ [FHHR09b]. In our PhD place example, TU Berlin wins with $(2 + 2 \cdot \alpha)$ points under Copeland $^{\alpha}$ voting (see Figure 1). For $\alpha = 1$, there are also two more co-winners (Oxford University and MIT).

Maximin Voting. Let $\#_{\min}(x) = \min\{\#(x, y) : y \in C \setminus \{x\}\}$ for $x \in C$ (recall that $\#(c, c')$ is the number of votes ranking candidate c higher than candidate c'). According to Maximin voting, a candidate c wins if she has the maximum value $\#_{\min}(c)$. In our PhD place example, TU Berlin, Oxford University, and MIT are all winners under Maximin voting. Clearly, winner determination using Maximin voting can be done in time polynomial in the input size. The *Maximin* concept originates from decision theory [Wal49, Sni08]. There are many other names for this voting protocol. For instance, Fishburn [Fis77] called it *Condorcet procedure* and Young [You77] used the name *Minimax function*.

We conclude this section with a remark on the relation between scoring protocols and Condorcet-related protocols. Condorcet [dC85] argued that there are elections whose Condorcet winner is not elected by any scoring protocol that awards more points to the first ranked candidate than to the second ranked one, and so forth; for example, this holds true for Borda voting [BF02]. The following example is due to Brams and Fishburn [BF02, Section 9.3] and shall illustrate this phenomenon. Suppose that there is an election on three candidates, a , b , and c , with seven votes cast as follows:

three votes $a \succ b \succ c$,
 two votes $b \succ c \succ a$,
 one vote $b \succ a \succ c$, and
 one vote $c \succ a \succ b$.

The Condorcet winner of this election is a . She beats both b and c by 4-to-3. However, any scoring protocol assigning strictly more points to a candidate placed 2nd than to a candidate placed 3rd makes b win. Indeed, Moulin [Mou91] showed that no positional scoring protocol fulfills the Condorcet principle.

3.3 Further Voting Protocols

In this section, we introduce two more commonly used voting protocols which require several stages to aggregate votes. We also discuss one additional issue concerning the election of multiple winners.

Plurality with Runoff. This voting protocol consists of two rounds. In the first round, it orders the candidates according to the number of votes in which they rank first; all candidates but the first two in this new order are eliminated from the original votes. In case that two or more candidates are tied to pass the first round, Conitzer et al. [CRX09] argued that a candidate c is a winner if and only if there exists a way to break ties in all steps such that c wins. In this survey, we adopt a specific tie-breaking rule: Let C_1 be the set of candidates that have the highest number of first positions, and let C_2 be the set of candidates that have the second-highest number of first positions.

- If $|C_1| = 1$ and $|C_2| = 1$, or $|C_1| = 2$, then go to the second round.
- If $|C_1| = 1$ and $|C_2| \geq 2$, then the candidate $c \in C_2$ who has the highest number of second positions stays. If $|C_1| \geq 3$, then the two candidates among C_1 with the highest numbers of second positions pass the first round. For both cases ($|C_1| = 1 \wedge |C_2| \geq 2$ or $|C_1| \geq 3$): If there are more than two candidates to pass the first round, then for tie-breaking the number of third positions is used, and so on. If, however, after $m - 1$ steps, still more than two candidates are tied, then all these candidates pass the first round.

In the second round, Plurality voting is applied to the input votes restricted to the candidates that pass the first round to elect a winner.

The second round can be omitted if in the first round there is a candidate who ranks first in more than half of the votes.

In our PhD place example, TU Berlin safely passes to the second round. However, ETH Zurich and MIT each rank first in one vote, and second in no votes, so we have to consider the votes where they rank third. MIT ranks third in two votes but ETH Zurich in only one vote, so MIT can stay for the second round. After eliminating the other candidates, TU Berlin and MIT are tied 2-to-2 in the second round, so they both are co-winners.

Variations of Plurality With Runoff voting are widely used in the presidential elections of many countries (such as Austria, Brazil, and France). It is criticized

for its so-called *no-show paradox* [Mou91], which means that sometimes it may be advantageous not to submit your vote. Let us see an example to better understand this paradox. Suppose that there are 100 votes on the candidates, a , b , and c , with

30 votes $a \succ c \succ b$,
 41 votes $b \succ a \succ c$, and
 29 votes $c \succ b \succ a$.

The winner according to Plurality with Runoff is b . However, if two of the voters who favor a abstain, then in the first round a will be eliminated and c beats b by 59-to-41 in the second round. While this does not make candidate a win, these votes do prefer candidate c to candidate b .

Single Transferable Voting (STV). To select a single winner, STV deletes the candidates ranked first in the fewest votes. This procedure is repeated until a candidate ranks first in more than half of the *restricted votes*—the votes without deleted candidates. By deleting some candidates, some originally lower ranked candidates can be *transferred* to a higher position. STV can take up to $m - 1$ stages with m being the total number of candidates. This happens if in each stage no candidate ranks first in more than half of the restricted votes. Note that if there are only three candidates, then STV for the single winner case is equivalent to Plurality with Runoff voting, and, hence, suffers from the same “no-show paradox”.

When using STV in our PhD place example, Oxford University and Tsinghua University will be first deleted from the votes: No vote ranks Oxford University or Tsinghua University as the first candidate. Then every candidate ranking behind Oxford University or Tsinghua University in the original votes will be transferred to a higher position:

Parameterized complexity:	$B \succ M \succ Z$
Salary:	$Z \succ M \succ B$
English usage:	$M \succ B \succ Z$
Cultural activities:	$B \succ Z \succ M$

In the next stage, we delete MIT and ETH Zurich from the remaining votes. Finally, the only candidate remaining, that is, TU Berlin, is the winner according to STV.

STV with some modifications is often used in political elections, for instance in Australia, Ireland, and New Zealand.

Obviously, the winner determination problem for Plurality with Runoff or STV can be solved in time polynomial in the input size.

Multi-Winner Protocols. Multi-winner elections come into play whenever one has to elect an assembly whose members need to be authorized to take decisions on behalf of the society. Hence, for a multi-winner voting protocol, it is important to elect an assembly (winner set) that represents the society adequately.

Although the protocols stated above can be easily modified to return a set of winners, for all of them except for STV this does arguably not lead to an appropriate choice of winners [BF02, LB11]. An alternative way is based on the concept of “misrepresentation”. Basically, in this model, each vote can assign a misrepresentation value to every candidate. The set of winners is selected from the candidates such that the total misrepresentation is minimized.

Borda voting is a natural example for a misrepresentation function: Every vote assigns a misrepresentation value of zero to his favorite candidate, a value of one to his second choice, a value of two to the third choice, and so on.

One natural approach for selecting winners is to choose a set of, say, k winning candidates such that the sum of misrepresentation values is minimized (*minimum sum*); another way is to minimize the maximum misrepresentation (*minimum*) [BEH⁺10, BF02]. In both cases, in the model suggested by Chamberlin and Courant [CC83] every candidate can represent an unlimited number of votes, that is, within a selected assembly a vote is always represented by an assembly member for whom its misrepresentation value is minimal. Since this may lead to the situation that different assembly members represent different numbers of votes, Chamberlin and Courant suggested to use weights as a way out. In contrast, the model suggested by Monroe [Mon95] requires that every assembly member represents about the same number of votes, that is, at most $\lceil n/k \rceil$ and at least $\lfloor n/k \rfloor$ for n votes and k winners.

Unfortunately, all four problem variants resulting from combining Chamberlin and Courant’s as well as Monroe’s approach with “minimax” or “minimum sum” optimization are already NP-hard for the basic Borda misrepresentation function [BSU11, LB11, PRZ08].

Parameterized complexity analysis with respect to the parameters “number of winners”, “total misrepresentation value”, “number of voters”, and “number of candidates” has been started only recently [BSU11].

4 Kemeny Voting

In this section, we provide a case study on different parameterizations of the voting problem KEMENY SCORE (which was mentioned in Section 3.2). The optimization problem behind KEMENY SCORE can also be seen as a natural combinatorial median finding problem: Given a multiset of rankings, find a ranking that is “closest” to the given rankings. Here, the distance measure is the so-called Kendall-Tau distance. Let (C, V) be an election and let l be a ranking over C . Then, the *score* of l is defined as

$$\sum_{v \in V} \text{KT-dist}(v, l),$$

where $\text{KT-dist}(v, l)$ denotes the *Kendall-Tau distance*. The Kendall-Tau distance between $v \in V$ and l is defined as

$$\text{KT-dist}(v, l) := \sum_{\{a, b\} \subseteq C} d_{v, l}(a, b),$$

where $d_{v,l}(a,b)$ is 0 when v and l rank a and b in the same relative order, and 1, otherwise. Formally, the corresponding decision problem is defined as follows:

KEMENY SCORE

Input: An election (C, V) and a non-negative integer k .

Question: Is there a ranking with score at most k ?

A *Kemeny consensus* l^* is a ranking with minimum score. The *Kemeny score* of a given election is the score of l^* .

The Kemeny score of our PhD place example (see Section 3) is sixteen. For instance, the ranking $B \succ O \succ M \succ Z \succ T$ and the ranking $O \succ M \succ B \succ T \succ Z$ each forms a Kemeny consensus. There are altogether eighteen different Kemeny consensuses. The reason is that most candidate pairs are tied 2-to-2 and both relative orderings of these two candidates contribute the same to the score. Every Kemeny consensus for our PhD place example realizes the cheaper relative ordering for all four non-tied candidate pairs (see Table 2 or Figure 1 in Section 3.2).

For small examples like this, a Kemeny consensus is easy to find. However, in practice one often has to deal with larger and more complicated instances. KEMENY SCORE is NP-hard, but in some applications exact solutions are required. Here, parameterized algorithmics comes into play. In the remainder of this section, we overview recent research concerning the parameterized complexity of KEMENY SCORE (see also Table 3).

4.1 Input and Output Parameterizations

Three parameters directly appear in the problem definition of KEMENY SCORE. The parameters “number n of votes” and “number m of candidates” are given by the input. The parameter “Kemeny score k ” is given by the solution of the problem.

Number n of Votes. KEMENY SCORE is NP-hard even for elections with only four votes [DKNS01a, DKNS01b]. This means that there is no hope for fixed-parameter tractability with respect to the parameter “number of votes”. To the best of our knowledge, NP-hardness for KEMENY SCORE with a constant odd number of votes is still open. On the contrary, NP-hardness for KEMENY SCORE with an unbounded odd number of votes has been shown by Bartholdi et al. [BTT89b].

Number m of Candidates. KEMENY SCORE becomes fixed-parameter tractable for the parameter “number of candidates”. This is easy to see: Try all possible $m!$ rankings over C , compute the corresponding scores, and check whether the minimum score is at most k . Note that, given an election with n votes and m candidates, one can compute the score of any ranking in $\mathcal{O}(n \cdot m \log m)$ time [KT06].

By a dynamic programming approach, one can improve the exponential part of the running time from $m!$ to 2^m [BFG⁺09, RS07]. The basic idea behind

Table 3. Parameterized complexity of KEMENY SCORE and two of its generalizations. In case of fixed-parameter tractability results, we only state the exponential parts of the corresponding running times if provided in the corresponding papers. “NP-h” means NP-hard. Results marked by (♣) follow from [DKNS01a, DKNS01b], (◇) follow from [KS10], (♠) follow from [MRS09], and (♡) follow from [BGKN11]. The remaining results are provided in [BFG⁺09]. Note that “?” means that the corresponding case remains open whereas “—” means that the corresponding parameter does not apply to the problem.

	KEMENY SCORE	with ties	incomplete votes
# votes n	NP-h for $n = 4$ (♣)	NP-h for $n = 4$ (♣)	NP-h for $n = 4$ (♣)
# candidates m	2^m	2^m	2^m
Kemeny score k	$2^{\mathcal{O}(\sqrt{k})}$ (◇)	1.76^k	$k! \cdot 4^k$
max. range r_m	32^{r_m}	$(3r_m + 1)! \cdot 2^{3r_m + 1}$	—
avg. range r_a	NP-h for $r_a \geq 2$	NP-h for $r_a \geq 2$	—
max. KT-dist d_m	$2^{\mathcal{O}(\sqrt{d_m})}$ (◇)	$(6d_m + 2)! \cdot 2^{6d_m + 2}$	NP-h for $d_m = 0$
avg. KT-dist d_a	$2^{\mathcal{O}(\sqrt{d_a})}$ (◇)	$2^{\mathcal{O}(d_a^2)}$ (♡)	NP-h for $d_a = 0$
$\bar{d} := k/n$	$2^{\mathcal{O}(\sqrt{\bar{d}})}$ (◇)	$2^{\mathcal{O}(\bar{d}^2)}$ (♡)	NP-h for $\bar{d} = 0$
above guarantee	FPT (♠)	?	?

the dynamic programming is to compute a Kemeny consensus for the elections restricted to subsets of candidates: The dynamic programming table contains a Kemeny consensus for each subset of candidates. We compute the entries for all subsets of size s beginning with $s = 1$. Then, we increase s until we get the entire candidate set. The initialization of the table is easy, because elections with only one candidate induce exactly one ranking. The recurrence behind the dynamic programming is as follows. Consider the computation of an entry for a subset $C' \subseteq C$. For each $c \in C'$, compute the score of the ranking beginning with c and concatenated with the Kemeny consensus for $C' \setminus \{c\}$ obtained from the dynamic programming table. Now, the entry for C' is a ranking with minimum score.

Kemeny Score k . The Kemeny score measures the “distance of the solution from the input votes”. The following two simple data reduction rules lead to a problem kernel with at most $2k$ votes and at most $2k$ candidates [BFG⁺09]. Herein, we call a pair of candidates $\{a, b\}$ *conflict pair* if there is one vote with “ $a \succ b$ ” and another vote with “ $b \succ a$ ” in the election.

Rule 1. *Delete every candidate that is not involved in any conflict pair.*

Rule 2. *If there are more than k identical votes, then return “yes” if the score of one of them is at most k ; otherwise, return “no”.*

The problem kernel obtained through Rules 1 and 2 already shows fixed-parameter tractability of KEMENY SCORE with respect to the parameter k . This can be improved by considering the conflict pairs. In this way, one obtains bounded search-tree algorithms which are much faster than an $\mathcal{O}^*((2k)!)$ -time⁷ brute-force strategy or an $\mathcal{O}^*(2^{2k})$ -time dynamic programming algorithm operating on the problem kernel. First, observe that the number of conflict pairs is at most k for every yes-instance [BFG⁺09]. A search-tree which decides for each conflict pair which of both orderings appears in a Kemeny consensus has size $\mathcal{O}(2^k)$. Considering “conflict triples” one obtains an improved algorithm with running time $\mathcal{O}(1.53^k + m^2 \cdot n)$ [BFG⁺09]. Further refined search-tree algorithms lead to search-tree sizes of $\mathcal{O}^*(1.403^k)$ [Sim09]. Besides search tree algorithms, further approaches were considered in the literature to solve KEMENY SCORE—yielding sub-exponential time fixed-parameter algorithms with respect to the parameter k [ALS09, FFL⁺10, KS10].

4.2 Structural Parameterizations

Depending on the voting protocols used, voting problems provide a large amount of interesting structural parameters. For KEMENY SCORE, we discuss the parameters “maximum range r_m of candidate positions”, “average range r_a of candidate positions”, “maximum KT-distance d_m between the input votes”, and “average KT-distance d_a between the input votes”. All four parameters are illustrated with the help of our PhD place example (see Table 1) in Figure 2. This section will be concluded by a brief discussion of an “above average parameterization” for KEMENY SCORE.

The parameters “maximum range r_m of candidate positions” and “average range r_a of candidate positions” both use a common concept called the *range* of a candidate. The *range* of a candidate c is defined as one plus the difference between her best and worst position.

Maximum Range r_m of Candidate Positions. The maximum range of candidate positions is the range of the candidate who has the maximum range. It seems plausible that instances with a bounded range of candidate positions are easier to solve. Indeed, using dynamic programming, one can solve KEMENY SCORE in $\mathcal{O}(32^{r_m} \cdot (r_m^2 \cdot m + r_m \cdot m^2))$ time [BFG⁺09].

Average Range r_a of Candidate Positions. Analogously to the maximum range, the average range r_a of candidate positions is the average range of all candidates. A small maximum range indicates instances which are easy to solve, while instances with a small average range of candidate positions remain hard. Even for instances with $r_a = 2$ KEMENY SCORE remains NP-hard [BFG⁺09]: Given a KEMENY SCORE instance (C, V, k) , one can construct an equivalent instance with average range 2 by adding $|C|^2$ many new candidates and putting them at

⁷ The notation $\mathcal{O}^*(\cdot)$ is similar to $\mathcal{O}(\cdot)$, but only states the superpolynomial part of the running time.

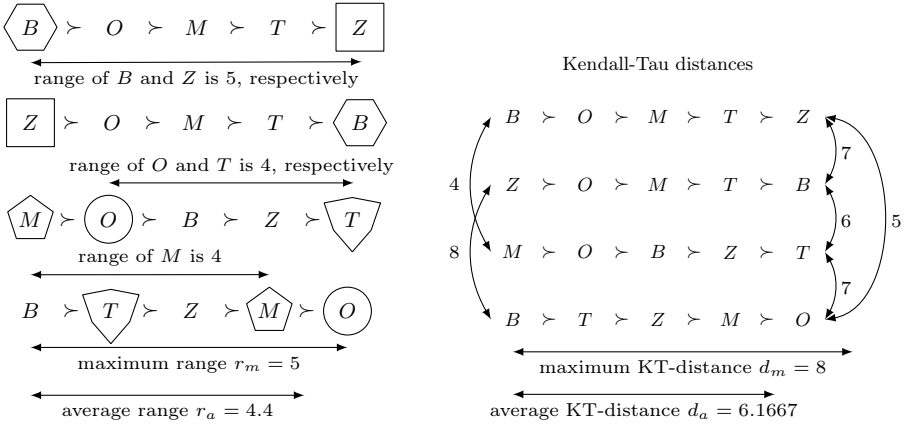


Fig. 2. Illustration of structural parameters for KEMENY SCORE. On the left we have our four votes from the PhD place example where the range of each candidate, that is, the difference between the worst and the best position is highlighted. The first vote is also one possible Kemeny consensus. On the right, we depict the KT-distances between every pair of input votes (written as labels on the arcs).

the end of every vote (for each vote in the same order). Each new candidate has a range of one and hence the average range is at most

$$\frac{|C| \cdot |C| + |C|^2}{|C|^2 + |C|} \leq 2.$$

Based on the Kendall-Tau distance, we discuss three further parameterizations.

Average KT-Distance d_a . The average KT-distance is formally defined as

$$d_a := \sum_{v,w \in V} \frac{\text{KT-dist}(v,w)}{n(n-1)}.$$

It measures “the average amount of variety in the votes”. In the first fixed-parameter algorithm with respect to parameter d_a [BFG⁺09], the authors basically observed that in every Kemeny consensus each candidate may only occur in a fixed range of positions whose size is bounded by d_a . Based on this observation, there is a dynamic programming algorithm that solves KEMENY SCORE in $\mathcal{O}(16^{d_a} \cdot (d_a^2 \cdot m + d_a \cdot m \log m \cdot n) + n^2 \cdot m \log m)$ time. This was improved by Simjour [Sim09] who developed a search tree algorithm with running time $\mathcal{O}^*(5.833^{d_a})$. Furthermore, Karpinski and Schudy [KS10] developed a subexponential fixed-parameter algorithm with running time $2^{\mathcal{O}(\sqrt{d_a})} + n^{\mathcal{O}(1)}$.

Besides fixed-parameter algorithms with respect to the parameter average KT-distance, data reduction rules were developed whose performance guarantee depends on the average KT-distance. Although no problem kernel in the

classical sense is known, the currently best upper bound on the number m of candidates is linear in the average KT-distance [BBN10]. This is achieved by applying polynomial-time data reduction. Note that the non-existing bound on the number n of votes does not harm too much, since KEMENY SCORE is fixed-parameter tractable with respect to m . More precisely, it was shown that exhaustive application of the following simple rule already yields a “partial problem kernel” [BBN10, BGKN11].

Rule 3. *If there is a candidate c such that there is no other candidate c' with $1/4 \cdot |V| \leq \#(c, c') \leq 3/4 \cdot |V|$, then remove c (and adjust the allowed score accordingly⁸).*

Exhaustive application of Rule 3 yields an equivalent instance of KEMENY SCORE with at most $16/3 \cdot d_a$ candidates [BBN10].

Maximum KT-Distance d_m . Clearly, fixed-parameter tractability for d_a also implies fixed-parameter tractability for the parameter “maximum KT-distance d_m between two input votes”. However its potentially larger values (compared to average KT-distance) allow for improvements in the algorithm. With slight modifications in the search tree algorithm for KEMENY SCORE parameterized by d_a , one can solve KEMENY SCORE in $\mathcal{O}^*(4.829^{d_m})$ time [Sim09]. Note that the subexponential fixed-parameter algorithm due to Karpinski and Schudy [KS10] for d_a also works for d_m .

Parameterizations Above Average k_{\min} . Mahajan and Raman [MR99] introduced “parameterization above guaranteed values” as a general form of parameterization. For KEMENY SCORE, a guaranteed value is a lower bound on the Kemeny score k , for instance

$$k_{\min} := \sum_{\{a,b\} \subseteq C} \min\{\#(a,b), \#(b,a)\}.$$

This is an obvious lower bound for k , because it is simply the sum of the minimum contributions for each candidate pair. A natural question is to parameterize above this guaranteed lower bound, that is, by the parameter “ $k - k_{\min}$ ”. Fixed-parameter tractability with respect to $(k - k_{\min})$ for KEMENY SCORE is implied by a parameter-preserving reduction from KEMENY SCORE to a weighted variant of DIRECTED FEEDBACK VERTEX SET [MRS09].

4.3 Ties and Incomplete Votes

In this section, we briefly discuss results obtained for two generalizations of KEMENY SCORE. In the first generalization, we modify our election model such that candidates may also be ranked equally, that is, we allow for ties. The second

⁸ For each candidate c' with $\#(c', c) > 3/4 \cdot |V|$, decrease the score by $\#(c, c')$; otherwise, decrease the score by $\#(c', c)$.

generalization is to allow for incomplete votes, that is, considering partial orders instead of linear orders (see Section 2 for a formal definition of incomplete votes). In contrast to the parameterization by “number of candidates”, which can also be used for both generalizations more or less without any modification (compare with Section 4.1), for most other parameterizations the situation changes when we consider the more general models.

Kemeny Score with Ties. In the KEMENY SCORE generalization KEMENY SCORE WITH TIES [Ail10, HSV05] one additionally allows that two candidates in a vote are ranked equally. Now, the term $d_{v,w}(a,b)$ expressing the contribution of the candidate pair $\{a,b\}$ to the KT-distance between two votes v and w is defined as

$$d_{v,w}(a,b) = \begin{cases} 2 & \text{if } (a \succ b \text{ in } v \text{ and } b \succ a \text{ in } w) \text{ or } (b \succ a \text{ in } v \text{ and } a \succ b \text{ in } w), \\ 0 & \text{if } a \text{ and } b \text{ are ordered in the same way in } v \text{ and } w, \text{ and} \\ 1 & \text{otherwise.} \end{cases}$$

There are slightly different models for the consensus of an election with ties in the literature: Hemaspaandra et al. [HSV05] allowed that the consensus can also have ties, while Ailon [Ail10] defined the consensus as permutation of candidates (without ties).

Betzler et al. [BFG⁺09] analyzed the parameterized complexity of KEMENY SCORE WITH TIES for the setting of Hemaspaandra et al. [HSV05]. With similar approaches as described in Section 4.1, one obtains a search tree of size $\mathcal{O}(1.76^k)$ as well as a polynomial-size problem kernel with respect to the parameter “Kemeny score k ” [BFG⁺09].

Concerning structural parameters such as maximum range r_m or average range r_a , one has to be careful when ties are allowed. Betzler et al. [BFG⁺09] used an intuitive concept where, similarly to the classical KEMENY SCORE, the range is defined as the difference between the best and the worst position. However, to make these positions in a vote with ties uniquely determined, the best position of a candidate is defined as the minimum number of candidates that are better than her and her worst position is defined as the maximum number of candidates that are better or equally ranked.

It is not obvious how to transfer the results for structural parameterizations with classical KEMENY SCORE to KEMENY SCORE WITH TIES. However, fixed-parameter tractability with respect to the parameter maximum range r_m of candidate positions can be obtained by an approach similar to the dynamic programming algorithm for classical KEMENY SCORE with respect to r_m [BFG⁺09]. Furthermore, when extending the problem by additionally assigning weights to candidates, the dynamic programming approach also covers the parameterization with maximum KT-distance d_m . The maximum range of candidate positions is bounded by $2 \cdot d_m$ for instances with candidate weights [BFG⁺09]. Finally, a partial kernelization with respect to the parameter average KT-distance can be transferred to KEMENY SCORE WITH TIES [BGKN11].

Kemeny Score with Incomplete Votes. In the KEMENY SCORE generalization KEMENY SCORE WITH INCOMPLETE VOTES [DKNS01a], the given votes are not required to be permutations of the entire candidate set, but of candidate subsets.⁹ In contrast to the votes, the Kemeny consensus is a permutation of all candidates. As a consequence, the term $d_{v,w}(a,b)$ expressing the contribution of the candidate pair $\{a,b\}$ to the KT-distance between two votes v and w is adjusted to

$$d_{v,w}(a,b) := \begin{cases} 0 & \text{if } \{a,b\} \not\subseteq C_v \text{ or } \{a,b\} \not\subseteq C_w \text{ or } v \text{ and } w \text{ agree on } a \text{ and } b, \\ 1 & \text{otherwise,} \end{cases}$$

where C_v contains the candidates occurring in vote v .

Since one can have non-trivial instances without “conflict pairs”, the branching approach for classical KEMENY SCORE does not apply to the parameterization with the Kemeny score k when we allow for incomplete votes. However, by a parameterized reduction to WEIGHTED FEEDBACK ARC SET, one obtains fixed-parameter tractability [BFG⁺09].

As to structural parameterizations, defining the range of candidate positions does not make sense. Furthermore, KEMENY SCORE WITH INCOMPLETE VOTES remains NP-hard even if the maximum KT-distance d_m between two input votes is zero, that is, there is no hope for fixed-parameter tractability with respect to the parameters average KT-distance d_a and maximum KT-distance d_m [BFG⁺09].

5 Types of Voting Problems

In this section, we review a number of voting problems and account for their computational complexity, both standard and parameterized. We start with the most immediate question in Section 5.1: Can a candidate win an election under a given voting protocol? In later sections we deal with more subtle voting problems, often rendering the considered problems already hard for scoring protocols. In Section 5.2, we consider possible winner determination in case of incomplete votes (partial orders instead of linear orders), then move on to the related problem of manipulating elections (Section 5.3), and eventually study questions of bribery, control, and optimal lobbying in Sections 5.4, 5.5, and 5.6.

5.1 Winner Determination

The most basic computational task in voting is the determination of a winner using a given voting protocol \mathcal{E} . Alternatively, we can ask whether a given candidate is a winner of an election under voting protocol \mathcal{E} :

\mathcal{E} WINNER DETERMINATION

Input: An election (C, V) and a distinguished candidate $p \in C$.

Question: Does p win the election under voting protocol \mathcal{E} ?

⁹ This only yields a subset of all possible partial orders.

Table 4. Parameterized complexity results for computationally hard winner determination problems. Considered parameters are the number m of candidates, the number n of votes, and the number k of modifications. For DODGSON SCORE, k denotes the number of swaps; for YOUNG SCORE, k denotes the number of remaining votes, while for DUAL YOUNG SCORE k denotes the number of deleted votes. The fixed-parameter tractability results for parameter m follow from integer linear programming formulations [BTT89b, You77] and Lenstra’s result on integer linear programming with a fixed number of variables [Len83]. Results marked by (Δ) follow from [BTT89b, You77], by (\clubsuit) from [FJL⁺10], by (\heartsuit) from [BGN10], and by (\spadesuit) from [RSV03].

Parameter	DODGSON SCORE	DUAL YOUNG SCORE	YOUNG SCORE
m	FPT (Δ)		FPT (Δ)
n	W[1]-hard (\clubsuit)		FPT ($\mathcal{O}^*(2^n)$) (Δ)
k	FPT ($\mathcal{O}^*(2^k)$) (\heartsuit)	W[2]-complete (\heartsuit)	W[2]-complete (\heartsuit , \spadesuit)

A voting protocol having nice properties but for which one cannot compute a winner in reasonable time is not useful in practice. While for some voting protocols such as positional scoring protocols, the computation of a winner can be easily achieved in polynomial time, for other voting protocols the computation of a winner is NP-hard. Famous voting protocols with NP-hard winner determination are listed in the survey by Chevaleyre et al. [CELM07]. This includes the voting protocols proposed by Banks, Dodgson, Kemeny, Slater, and Young.¹⁰

In Table 4, we list parameterized complexity results for DODGSON SCORE, DUAL YOUNG SCORE, and YOUNG SCORE with respect to several parameterizations.

5.2 Possible and Necessary Winner

Possible Winner. In standard voting scenarios, one typically assumes that voters provide their preferences as linear orders. To determine a winner, the given linear orders are aggregated according to a voting protocol. However, in many realistic settings, the voters may provide partial orders only [KL05]. This directly leads to the POSSIBLE WINNER problem which, given a set of incomplete votes, asks whether a specific candidate can still become a winner if one extends the votes to linear orders (see Section 2 for detailed information on partial orders and their extensions).

Let us go back to the student from Section 3 who decides to do his PhD research at the TU Berlin. At the enrollment, it happens that there is a MY FAVORITE PROFESSOR evaluation among four candidate professors: Prof. Bosch (B),

¹⁰ Banks [Ban85] and Slater [Sla61] are two comparison-based voting protocols. They both work on the majority graph of a given election (see Definition 5) and are closely related to graph problems restricted to tournaments [CH00, Woe03]. A tournament is a directed graph with exactly one arc between any two vertices. See Woeginger [Woe03] and Hudry [Hud04] for the computation of Banks winners, and Charon and Hudry [CH00] and Conitzer [Con06] for the computation of Slater winners.

Prof. Geiger (G), Prof. Hertz (H), and Prof. Zuse (Z).¹¹ Until now, only sixteen students have participated in the evaluation. Five of them like Prof. Bosch as much as Prof. Geiger (in the subsequent example expressed by $\{B, G\}$), followed by Prof. Hertz and then by Prof. Zuse. Another five students favor Prof. Hertz over Prof. Zuse, followed by Prof. Bosch, while ranking Prof. Geiger as the least-liked candidate. The remaining six students prefer Prof. Geiger and Prof. Zuse to Prof. Hertz. Their least favorite professor is Prof. Bosch. The current state of the evaluation is as follows:

Five students with $\{B, G\} \succ H \succ Z$,
 five students with $H \succ Z \succ B \succ G$, and
 six students with $\{G, Z\} \succ H \succ B$.

Obviously, the preferences of the students are not all linear orders. Hence, instead of determining the best ranked professor, we are interested in determining the *possible winners* of the election. For instance, to ask whether Prof. Hertz is a possible winner under Borda voting in the above evaluation is to determine whether there are extensions of students' preferences such that Prof. Hertz becomes a winner. In our example, such extensions exist: If the first five students submit the linear order $B \succ G \succ H \succ Z$, three of the last six students submit $G \succ Z \succ H \succ B$ and the other three students submit $Z \succ G \succ H \succ B$, then Prof. Hertz (26 points) becomes a winner under Borda voting. However, to determine whether a distinguished candidate is a possible winner in an election using Borda voting is NP-complete [XC11] (also see Table 5).

Formally, POSSIBLE WINNER for a given voting protocol \mathcal{E} is defined as follows:

\mathcal{E} POSSIBLE WINNER

Input: An election (C, V) with the multiset $V = \{v_1, \dots, v_n\}$ of incomplete votes represented as partial orders on C , and a distinguished candidate $p \in C$.

Question: Is there a multiset $V' = \{v'_1, \dots, v'_n\}$ of votes over C , such that each vote v'_i extends v_i and p wins the election (C, V') under voting protocol \mathcal{E} ?

The motivation behind POSSIBLE WINNER is that it might be impossible for the voters to provide a complete ranking because, for instance, the set of candidates is too large. Another reason can be that not all voters might have given their rankings yet during the aggregation process, or new candidates might be introduced after some voters already have given their rankings (see also [CLM⁺11]). Moreover, one often has to deal with incomplete votes due to two or more candidates not being comparable, because of lack of information or other reasons. Hence, the study of incomplete voting profiles is natural and essential.

¹¹ Historical note: Only Hans Geiger and Gustav Hertz were professors at TU Berlin whereas Carl Bosch and Konrad Zuse were students at TU Berlin.

Table 5. Summary of (parameterized) complexity results for the NP-complete [XC11] POSSIBLE WINNER using several common voting protocols. Parameters considered are “the number m of candidates”, “the number n of votes”, “the total number s of undetermined candidate pairs”, and “the maximal number u of undetermined candidate pairs in a vote”. Note that the fixed-parameter tractability results for parameter s hold for all voting protocols whose WINNER DETERMINATION problem can be solved in time polynomial in the input size. Fixed-parameter tractability results for parameter m are again due to Lenstra’s result on integer linear programming with a fixed number of variables [Len83]. Results marked with (\clubsuit) come from [XC11], those with (\heartsuit) come from [BHN09]. Note that “?” means that the corresponding case remains open and para-NP-c means that the problem remains NP-complete even for constant parameter values.

Parameter	Borda	k -Approval	Copeland $^\alpha$
m	FPT	FPT	FPT
n (\heartsuit)	para-NP-c	para-NP-c	?
s (\heartsuit)	$\mathcal{O}^*(1.82^s)$	$\mathcal{O}^*(2^s)$	$\mathcal{O}^*(2^s)$
u (\clubsuit)	para-NP-c	para-NP-c	para-NP-c

Again, we survey standard and parameterized computational complexity results for POSSIBLE WINNER under various voting protocols. Notably, although WINNER DETERMINATION problems are straightforward for (most) scoring protocols, POSSIBLE WINNER is already computationally hard for simple scoring protocols such as k -Approval. Table 5 lists the results together with references to the literature.

Due to the way how k -Approval assigns points to candidates, two further structural parameters immediately pop up in the study of k -APPROVAL POSSIBLE WINNER: The “number k of approvals in each vote” and the “number $k' = m - k$ of disapprovals in each vote” with m being the total number of candidates. However, k -APPROVAL POSSIBLE WINNER is already NP-complete for any constant number $k \geq 2$ [XC11]. This motivates a multivariate complexity analysis [Fel09, Nie10] with respect to the combined parameter number n of votes and number k (k') of candidates to whom a voter gives one (zero) point. Parameterized complexity results for k -APPROVAL POSSIBLE WINNER are summarized in Table 6.

There are many interesting open questions concerning \mathcal{E} POSSIBLE WINNER. In the following we just mention a few:

- Until now, existing studies [BD10, BHN09, Wal07, XC11] on POSSIBLE WINNER consider only scoring protocols as well as some comparison-based protocols that are computationally efficient (polynomial time solvable) for

Table 6. Parameterized complexity results of k -APPROVAL POSSIBLE WINNER, where t denotes the number of incomplete votes in an election, k denotes the number of ones assigned in the k -Approval voting, while k' denotes the number of zeros assigned in the k -Approval voting. Results marked with \clubsuit come from [XC11], while \heartsuit marks results from [Bet10b].

Parameter	Results	Remarks
k (\clubsuit)	NP-complete	For any fixed $k \geq 2$
(t, k') (\heartsuit)	FPT	$\mathcal{O}^*(\min\{2^{t^2 k'}, 2^{t k'} \cdot (t k')^{k'}\})$
(t, k) (\heartsuit)	FPT	Super-exponential kernel

WINNER DETERMINATION, since if the \mathcal{E} WINNER DETERMINATION is computationally hard, then \mathcal{E} POSSIBLE WINNER is also computationally hard. It would be interesting to see whether the fixed-parameter tractability results for KEMENY SCORE, DODGSON SCORE, YOUNG SCORE, or DUAL YOUNG SCORE still hold for POSSIBLE WINNER where incomplete votes are given.

- As we have seen in Table 5, the parameter “total number s of undetermined candidate pairs” leads to fixed-parameter tractability; however, s may be very large for some scenarios. On the contrary, POSSIBLE WINNER is already NP-complete for Borda, k -Approval, and Copeland ^{α} voting even if the maximal number u of undetermined candidate pairs in a vote is a constant [XC11]. This motivates further parameterizations concerning incomplete votes of an election. For example, it would be interesting to know whether POSSIBLE WINNER is fixed-parameter tractable with respect to the parameter “average/maximum number of undetermined candidate pairs in which a candidate is involved”.

Necessary Winner. Finally, we mention in passing that, in addition to the POSSIBLE WINNER problem, there is also the NECESSARY WINNER problem, which asks whether a given distinguished candidate is a winner in *all* extensions of the given votes. As a rule of thumb, it appears that the NECESSARY WINNER problem is computationally easier than the POSSIBLE WINNER problem. For example, NECESSARY WINNER can be solved in polynomial time for scoring protocols as well as some other protocols such as Plurality with Runoff, Maximin voting, and Bucklin¹², while POSSIBLE WINNER is NP-complete for these voting protocols [XC11]. We refer to the literature [KL05, PRVW11, XC11] for more details.

¹² Bucklin voting is a hybrid voting protocol. In a nutshell, it combines k -Approval with Majority voting. Majority voting is similar to Plurality with the additional constraint that the candidate who has a score of more than half of the number of votes wins. See Xia and Conitzer [XC11] for a definition and more in-depth explanation.

5.3 Manipulation

Manipulation is a voting scenario where a *coalition of voters* casts their votes in an insincere way such that they end up better off than voting honestly. We illustrate such a situation with the help of the MY FAVORITE PROFESSOR example. Suppose that the election has

- five students with $B \succ G \succ H \succ Z$,
- five students with $H \succ Z \succ B \succ G$,
- three students with $G \succ Z \succ H \succ B$, and
- three students with $Z \succ G \succ H \succ B$.

Under Borda voting, Prof. Hertz (26 points) wins the election. Suppose that the last three students in the above election know the votes of all other thirteen students.¹³ They want to make their favorite candidate, Prof. Zuse, win the election. Hence they form a *coalition* and try to *manipulate* the election result by casting their own votes contrary to their actual preferences. Although they all prefer Prof. Geiger and Prof. Hertz to Prof. Bosch, by submitting

- two votes $Z \succ G \succ B \succ H$ and
- one vote $Z \succ B \succ G \succ H$,

together with the other thirteen votes, Prof. Zuse will indeed become the Borda winner with 25 points instead of Prof. Hertz with 23 points.

For manipulation, we assume that the voters of the coalition know about all the votes of the sincere voters. The coalition uses *strategic voting* to achieve their goal of letting their favorite candidate win. Formally, the decision problem MANIPULATION for any voting protocol \mathcal{E} is defined as follows:

\mathcal{E} MANIPULATION

Input: An election (C, V) , a coalition size $k \in \mathbb{N}$ encoded in unary alphabet, and a distinguished candidate $p \in C$.

Question: Is there a multiset V' of at most k votes on C such that p is the winner according to \mathcal{E} in $(C, V \cup V')$?

The \mathcal{E} MANIPULATION problem can be considered as a special case of the \mathcal{E} POSSIBLE WINNER problem: The non-manipulative votes are linear orders and the manipulative votes are totally empty. Hence, any hardness result on \mathcal{E} MANIPULATION is also valid for \mathcal{E} POSSIBLE WINNER.

A voting protocol is *strategy-proof* if manipulation is never beneficial for any voter or coalition of voters. A famous result of Gibbard and Satterthwaite [Gib73, Sat75] states that a *resolute*¹⁴, *surjective*¹⁵, and strategy-proof

¹³ This is rarely the case in practice. However, it allows for a worst-case analysis.

¹⁴ A voting protocol is *resolute* if there is always exactly one winner for an election.

¹⁵ A voting protocol is *surjective* if every candidate has a chance of winning.

voting protocol is dictatorial¹⁶. Bartholdi et al. [BTT89a] suggested using computational hardness to “resist” manipulations in an election: The idea is that if a voting protocol can be manipulated in principle, but it is computationally intractable to decide whether it is possible to cast the votes to achieve a desired result, then this voting protocol is unlikely to be manipulated in practice. In particular, Bartholdi et al. [BO91, BTT89a] focused on the special case of having a coalition of size one: After obtaining polynomial-time solvability results for manipulation under a set of common voting protocols including Plurality, Borda, Maximin and Copeland [BTT89a], Bartholdi and Orlin [BO91] showed that STV MANIPULATION is NP-hard even for a single manipulator. However, Conitzer et al. [CSL07] showed fixed-parameter tractability with respect to “the number m of candidates” for STV MANIPULATION with a coalition of size one. The corresponding algorithm runs in $\mathcal{O}^*(1.62^m)$ time. Recent studies [BNW11, DKNW11] show that BORDA MANIPULATION is already NP-complete for a coalition of size two. When parameterized by “the number of candidates”, BORDA MANIPULATION is fixed-parameter tractable [BHN09]. A further parameter is derived from so-called “instance tightness”, again yielding fixed-parameter tractability [BNW11].

Since \mathcal{E} MANIPULATION is a special case of \mathcal{E} POSSIBLE WINNER, some open computational hardness questions stated in Section 5.2 can also be transformed to the context of \mathcal{E} MANIPULATION.

For STV MANIPULATION with one manipulator, there is a fixed-parameter algorithm with respect to “the number of candidates” [CSL07]. Naturally, it would be interesting to know whether this also holds for two or more manipulators.

5.4 Bribery

As the name suggests, *bribery* is another attack on elections, where the briber “pays” some voters to have them change their votes in order to reach a desired outcome [FHH09]. Typically, the briber has a budget. The basic question with respect to bribery is whether the briber can achieve his goal without exceeding his budget.

There are different settings of bribery: Besides varying prices for different voters, one relaxes the notion of votes by allowing arbitrary relations instead of linear orders [Fal08, FHHR09b]. In addition, there are more fine-grained models such as paying for specific operations. For instance, in SWAP BRIBERY [EFS09], one is only allowed to perform *swaps* of two neighboring candidates in a vote. Formally, a swap in some vote $v \in V$ is a triple (v, c_1, c_2) where $\{c_1, c_2\} \subseteq C, c_1 \neq c_2$. Applying a swap (v, c_1, c_2) , that is, exchanging the positions of c_1 and c_2 in the vote v , is *admissible* when c_1 and c_2 are neighbors in v . A sequence of swaps is called admissible when the application of the swaps in the given ordering is admissible in each case. The decision problem is defined as follows:

¹⁶ *Dictatorial* means that there exists a voter who always decides what the outcome of an election shall be.

\mathcal{E} SWAP BRIBERY

- Input:* An election (C, V) , a distinguished candidate $p \in C$, a budget $\beta \in \mathbb{N}$, and a cost function $c : V \times C \times C \rightarrow \mathbb{N}$.
- Question:* Is there an admissible sequence Γ of swaps with $\sum_{s \in \Gamma} c(s) \leq \beta$ such that p wins the election under voting protocol \mathcal{E} after having applied the swaps as given by Γ ?

For our MY FAVORITE PROFESSOR example (see Section 5.3 for the complete list of student votes), we already know that Prof. Hertz wins under Borda voting. Suppose that a fan of Prof. Bosch knows all the votes of the students. His goal is to make Prof. Bosch win the election via swap bribery. A single swap costs one Euro. However, he is only willing to pay at most four Euros. Now the question is whether, without exceeding his budget, the fan of Prof. Bosch can bribe some students and let them *swap* neighboring candidates in their votes such that Prof. Bosch wins the election. If he bribes the three students with identical original vote $G \succ Z \succ H \succ B$ and one student whose original vote is $Z \succ G \succ H \succ B$, and lets them each swap the two neighboring candidates H and B in their votes, then he can make B (Prof. Bosch with 26 points) win the election. Each of the four swaps has a cost of one Euro, so the budget (four Euros) is not exceeded.

Table 7 shows some computational complexity results for k -APPROVAL SWAP BRIBERY. Classical complexity results are given by Elkind et al. [EFS09], while the parameterized results are provided by Dorn and Schlotter [DS12]. It should be mentioned that \mathcal{E} POSSIBLE WINNER can be seen as a special case of \mathcal{E} SWAP BRIBERY, where the price of any determined candidate pair is one, swapping two *undetermined* neighboring candidates¹⁷ has cost zero, and the budget is zero. So hardness results on \mathcal{E} POSSIBLE WINNER are also valid for \mathcal{E} SWAP BRIBERY for some restricted scenarios.

Restricting the allowed operations such that each swap must involve the distinguished candidate leads to SHIFT BRIBERY [EFS09]. As for the parameterized complexity analysis of this scenario, we refer to a recent study by Schlotter et al. [SEF11].

In *microbribery* [FHHR09b], a briber can invert the relative order of any two candidates in a vote for a given price. Typically, this leads to votes which are no longer linear orders. For example, inverting the relative order of a and c in a vote $a \succ b \succ c$ results in three pairwise comparisons: $a \succ b$, $b \succ c$, and $c \succ a$.

Elkind and Faliszewski [EF10a] initiated research on another aspect of bribery which concerns *campaign management*. There, bribing voters means, for instance, investing in advertisement for a specific candidate. They argued that such kind of campaign can strongly influence the outcome of an election. This has applications in political elections or product marketing. Schlotter et al. [SEF11] studied both classical and parameterized complexity regarding two specific cases of campaign management, *shift bribery* and *support bribery*, for several voting protocols.

¹⁷ Recall that two neighboring candidates are called *undetermined* if they are not comparable in the incomplete vote.

Table 7. Parameterized complexity results of k -APPROVAL SWAP BRIBERY [DS12]. The parameters are “the budget β ”, “the number n of votes”, “the number m of candidates”, and “the number k of approved candidates in a vote”. Note that k -APPROVAL SWAP BRIBERY is already NP-complete for $k = 2$ due to its close relationship to POSSIBLE WINNER [BD10, DS12, EFS09]. With respect to the parameter m , the fixed-parameter tractability result holds not only for k -Approval but indeed for a wide range of voting protocols including Copeland ^{α} and Maximin [DS12].

Parameter	Results	Remarks
β	W[1]-hard for $n = 1$	Reduction from MULTI-COLORED CLIQUE
k	W[1]-hard	Reduction from CLIQUE
m	FPT for constant k	Integer linear programming
n	FPT for constant k	Color-coding
(β, n)	FPT	Kernel with $n^2\beta^2$ candidates and $n^2\beta$ votes
(β, n, k)	FPT	Kernel with $(n + k)\beta$ candidates and $n^2\beta$ votes

Destructive bribery, that is, using bribery to prevent one candidate from winning, is NP-hard for Copeland ^{α} and Maximin voting [FHH09, FHH11, FHHR09b]. Until now, parameterized complexity aspects in this context seem to be unexplored, presenting good opportunities for new research.

5.5 Control

To *control* an election, an external agent, somewhat misleadingly called the *chair* in the literature, can change the *election structure* to reach certain goals. For example, a typical question is whether the chair can make his favorite candidate a winner by deleting some candidates. Going back to our MY FAVORITE PROFESSOR example, using Copeland^{0.5} also results in selecting Prof. Hertz as the most favorite professor (2 points). If a fan of Prof. Bosch who wants to influence the election is in the election committee and somehow manages to disqualify Prof. Zuse from the election, then all three remaining candidates become (co)-winners (1 point) according to Copeland^{0.5}.

Actually, there are many different types of control including *adding* or *deleting* candidates or votes [BTT92]. Furthermore, one distinguishes between *constructive control* (CC), where the chair aims at making a distinguished candidate a winner, and *destructive control* (DC), where the chair wants to prevent a distinguished candidate from winning [HHR07]. In the following, we define the \mathcal{E} CONSTRUCTIVE CONTROL VIA ADDING CANDIDATES (\mathcal{E} CC-AC).

\mathcal{E} CC-AC

Input: Two disjoint sets C, D of candidates, a multiset V of votes over $C \cup D$, a distinguished candidate $p \in C$, and a non-negative integer k .

Question: Is there a subset $D' \subseteq D$ of candidates with $|D'| \leq k$ such that p is the winner in the election $(C \cup D', V)$ according to voting protocol \mathcal{E} ?

Three more types of constructive control problems, that is, via deleting candidates, via adding votes, and via deleting votes, can be defined analogously: For the case of adding votes, we are given a multiset of votes from which we can select additional votes in order to change the outcome of an election. The decision problems of destructive control via adding or deleting candidates or votes can be defined accordingly: Instead of making the distinguished candidate a winner, destructive control aims at precluding the distinguished candidate from winning.

The investigation of the computational complexity of control problems goes back to Bartholdi et al. [BTT92]. Since then, there has been a series of publications [BTT92, FHHR09b, HHR07] which provides a complete picture of the classical computational complexity for 22 basic types of control. These papers cover standard voting systems such as Plurality, Condorcet, and Copeland $^\alpha$ for all rational values of α in the range of $[0, 1]$. For example, one of the voting protocols that can be used to determine the winner of an election in time polynomial in the input size and is NP-hard for all standard types of constructive control is Copeland $^{0.5}$ [FHHR09b].

Hemaspaandra et al. [HHR09] showed that so-called *hybrid* elections can lead to stronger resistance results for electoral control. Further work looks into control for two specific hybrid systems combining Approval voting and systems based on linear preferences [EF10b, ENR09, EPR10, ER10].

A closely related problem introduced by Elkind et al. [EFS10a] is cloning, where one only allows for adding candidates that are “similar”¹⁸ to one of the existing candidates. Moreover, Chevaleyre et al. [CLM⁺11] investigated the question whether a candidate can become a winner by adding “arbitrary” candidates.

Recently, Faliszewski et al. [FHH11] introduced the extended scenario of “multi-mode control attacks”, that is, the chair is allowed to use various kinds of attacks like deleting candidates and adding votes simultaneously.

Table 8 lists some parameterized complexity results for eight different kinds of control: CONSTRUCTIVE CONTROL VIA ADDING CANDIDATES (CC-AC), CONSTRUCTIVE CONTROL VIA DELETING CANDIDATES (CC-DC), CONSTRUCTIVE CONTROL VIA ADDING VOTES (CC-AV), CONSTRUCTIVE CONTROL VIA DELETING VOTES (CC-DV), and their destructive control (DC) versions.

Table 9 shows some results on control of elections employing Copeland $^\alpha$. Considered parameters are the number m of candidates and the number n of votes. Furthermore, there are also results concerning parameterized complexity for Copeland $^\alpha$, $\alpha = 1$, with respect to non-standard parameters like “feedback arc set size of the majority graph” [BBNU11].

We conclude this section with a few interesting research directions. The parameterized complexity of many scoring protocols, such as Borda, seems to be unexplored. Multi-mode control as proposed by Faliszewski et al. [FHH11] seems a natural candidate for a multivariate complexity study. For instance, the problem whether a distinguished candidate can win by deleting k candidates and

¹⁸ Here, a candidate c_1 is called similar to another candidate c_2 if for each vote, candidates c_1 and c_2 have the same relative position to any other candidate.

Table 8. Control-related (parameterized) complexity results. All W-hardness results are with respect to the output parameter. For example, PLURALITY CC-AC is W[2]-hard with respect to the number of added candidates. Results marked with (♠) come from [BTT92], those with (◇) come from [HHR07], those with (♣) come from [LFZL09], those with (□) come from [LZ10], results marked with (♥) come from [BU09], those marked with (△) come from [FHHR09b], those marked with (○) come from [EF10b, EFPR11, ER10], and those marked with (▽) come from [BGN10]. The W[2]-completeness result of DUAL YOUNG SCORE holds for CONDORCET CC-DV because they are equivalent. Any entry labeled “P” means polynomial-time solvability. “/” means either that we are not aware of any meaningful parameterized complexity results or that it is irrelevant. For example, in CONDORCET CC-AC, the chair can never make a non-winning candidate win the election by adding some additional candidates [BTT92]. W[t]-h stands for W[t]-hard with $t = 1$ or $t = 2$; W[2]-c stands for W[2]-complete. Recall that Bucklin voting is a hybrid voting protocol which combines k -Approval with Majority voting [EF10b, XC11]. Fallback [BS09] voting combines Bucklin with Approval voting. Here, Approval voting, slightly different from k -Approval, allows each voter to approve of an arbitrary number of candidates.

	Plurality	Condorcet	Maximin	Copeland ^α	Bucklin/Fallback
CC-AC	W[2]-h (♠)	/	W[2]-h (□)	W[2]-c (♥)	W[2]-h (○)
CC-DC	W[2]-h (♥)	P (♠)	/	W[2]-c (♥)	W[2]-h (○)
CC-AV	P (♠)	W[1]-h (♣)	W[1]-h (□)	/	W[2]-h (○)
CC-DV	P (♠)	W[2]-c (▽)	W[1]-h (□)	/	W[2]-h (○)
DC-AC	W[2]-h (◇)	P (◇)	/	P (△)	W[2]-h (○)
DC-DC	W[1]-h (♥)	/	/	P (△)	W[2]-h (○)
DC-AV	P (◇)	P (◇)	W[1]-h (□)	/	P (○)
DC-DV	P (◇)	P (◇)	W[1]-h (□)	/	P (○)

adding k' votes under Copeland¹ is NP-hard [FHH11]; it is an open question whether this NP-hard problem is fixed-parameter tractable with respect to the combined parameter (k, k') .

Table 9. Parameterized complexity results on control of elections using Copeland^α. We use (♣) to denote the results from [FHHR09b] and (♥) to denote the results from [BU09]. Results on control by adding (deleting) candidates with a bounded number of added (deleted) candidates as well as on control by adding (deleting) votes with a bounded number of added (deleted) votes follow from brute-force enumeration of all possible subsets of candidates of votes. For the case of candidate control with a bounded number of votes, the fixed-parameter algorithms are based on Lenstra’s integer linear programming result [Len83].

	AC	DC	AV	DV
# candidates m	FPT (♣)	FPT (♣)	FPT (♣)	FPT (♣)
# votes n	NP-c (♥)	NP-c (♥)	FPT (♣)	FPT (♣)

5.6 Lobbying

Sometimes we do not only vote on one but on multiple issues at the same time. A corresponding voting procedure can be very simple: Approve or disapprove of each issue. Formally, a *multi-issue election* for m issues and n voters is an $n \times m$ binary matrix

$$W = \begin{pmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,m} \\ \vdots & \vdots & \ddots & \vdots \\ w_{n,1} & w_{n,2} & \cdots & w_{n,m} \end{pmatrix} \in \{0, 1\}^{n \times m}.$$

An entry $w_{i,j}$ of W represents voter i 's opinion on issue j : 0 stands for disapproval; 1 stands for approval.

Given a multi-issue election and desired outcomes for each issue, Christian et al. [CFRS07] studied how hard it is to “lobby” some voters optimally, that is, to persuade the minimum number of voters to change their votes such that each issue has a majority of voters with values equal to the desired outcome. The formal definition of OPTIMAL LOBBYING is as follows:

OPTIMAL LOBBYING

Input: A multi-issue election $W \in \{0, 1\}^{n \times m}$, a non-negative integer k , and a size m target vector $x \in \{0, 1\}^m$.

Question: Can W be transformed into a new matrix $W' \in \{0, 1\}^{n \times m}$ by editing entries in at most k different rows such that for each column j there is a strict majority of rows with value x_j ?

If OPTIMAL LOBBYING can be shown to be computationally intractable, then potential attackers may not succeed in influencing the outcome of a multi-issue election via lobbying in reasonable time. This is what Bartholdi et al. [BTT89b] and Faliszewski et al. [FHH10] meant by using complexity to protect elections. But what about more restricted scenarios, that is, what about parameterized complexity analysis? To the best of our knowledge, Christian et al. [CFRS07] started the parameterized complexity analysis of voting problems concerning lobbying in multi-issue elections. They showed that OPTIMAL LOBBYING is W[2]-complete with respect to the number of votes to be changed. The idea of the proof of this result is shown in Section 6.6. Erdély et al. [EFG⁺09] further extended OPTIMAL LOBBYING to a probabilistic setting and, in particular, provided several results on fixed-parameter tractability and W[2]-completeness. Finally, we remark that OPTIMAL LOBBYING is fixed-parameter tractable with respect to the number of issues, since it can be easily transformed into an integer linear program with a fixed number of variables. See also Section 6.4 for more details.

6 Parameterized Techniques

In this section, we overview different techniques for investigating fixed-parameter tractability which already have been successfully applied in the area of voting. We

start with some techniques for designing fixed-parameter algorithms and close with a general technique to obtain intractability results. Each technique will be accompanied by an example. Although these standard techniques “cover” most results so far, there are further approaches to obtain fixed-parameter tractability results in the context of voting [ALS09, FFL⁺10, KS10].

6.1 Search Trees

A search tree algorithm identifies a “small subset” of the input instance such that at least one part of the subset is part of a solution. Then, it branches over all possible parts of this small subset to fix it as part of the solution. This procedure is repeated in a recursive manner until the whole solution has been found. In the context of fixed-parameter algorithms, the identification of the subset is done in polynomial time and the search tree size is bounded by some function only depending on the parameter.

For instance, in the case of KEMENY SCORE (see Section 4.1) a simple search tree algorithm identifies as small subset the two possible orderings a candidate pair can have in the solution. Since one can show that for at most k many candidate pairs, where k denotes the Kemeny score of the election, the ordering is not yet clear, the search tree size is bounded by $\mathcal{O}(2^k)$.

Next, we briefly discuss a search tree approach that applies to POSSIBLE WINNER for arbitrary voting protocols [BHN09]. For every undetermined candidate pair, say $\{a, b\}$ from incomplete vote v , branch into the following two possible cases: Either add $a \succ b$ to v or add $b \succ a$ to v . If an option violates the transitivity of v , then discard the corresponding branch in the search tree. The search tree size is at most $\mathcal{O}(2^s)$, where s is the total number of undetermined candidate pairs, implying that for every voting protocol with polynomial-time winner determination, POSSIBLE WINNER is fixed-parameter tractable with respect to the parameter s .

Similarly to the improved search tree for KEMENY SCORE (see Section 4.1), one gains a refined fixed-parameter algorithm for POSSIBLE WINNER through considering undetermined triples instead of pairs combined with a network flow construction [BHN09].¹⁹ As a consequence, for a specific class of scoring protocols, including k -Approval and Borda, POSSIBLE WINNER can be decided in $O(1.82^s \cdot (nm^2 + s^2))$ time, where s is the total number of undetermined pairs.

6.2 Kernelization

Problem Kernels. Recall from Section 2 that a problem kernel can be seen as an equivalent instance whose size is bounded by a function in the parameter and which can be computed by polynomial-time preprocessing (so-called data reduction rules) [Bod09, GN07].

¹⁹ Indeed, techniques based on network flows are used in several other voting contexts to derive polynomial-time solvability for special cases or as part of a fixed-parameter algorithm (see, for example, [BD10, BHN09, DS12, FHHR09b]).

For instance, by exhaustive application of Rules 1 and 2 from Section 4.1 one gets a problem kernel with at most $2k$ votes and at most $2k$ candidates for KEMENY SCORE.

Dorn and Schlotter [DS12] developed a kernelization algorithm that constructs a problem kernel with $\mathcal{O}(n^2 \cdot \beta)$ votes and $\mathcal{O}(n^2 \cdot \beta^2)$ candidates for k -APPROVAL SWAP BRIBERY, where n denotes the number of votes and β denotes the budget. Kernelization algorithms have also been developed for k -APPROVAL POSSIBLE WINNER: Problem kernels for POSSIBLE WINNER with respect to the combined parameters (t, k) as well as (t, k') have been obtained by data reduction rules [Bet10a, Bet10b], where t denotes the number of incomplete votes, k denotes the number of one-point positions, and k' denotes the number of zero-point positions. For (t, k') there is a kernel with $\mathcal{O}(t \cdot k'^2)$ candidates and $\mathcal{O}(t^2 \cdot k'^2)$ votes, while for (t, k) there is a superexponential-size kernel that shows fixed-parameter tractability.

No Polynomial Kernel. A natural question for a fixed-parameter tractable problem is: How small can a corresponding problem kernel be? In particular, can we expect to derive a polynomial-size kernel for every fixed-parameter tractable problem? To answer this, Bodlaender et al. [BDFH09] and Fortnow and Santhanam [FS11] introduced a general framework; also see the surveys by Bodlaender [Bod09] and Misra et al. [MRS11]. The basic idea is that if a parameterized version of an NP-complete problem has a so-called “composition algorithm”, then it does not admit a polynomial-size problem kernel, under some widely believed complexity assumptions. Furthermore, such lower-bound results can be transferred to other problems by so-called “polynomial parameter transformations” [BTY11].

For instance, Fellows et al. [FJL⁺10] showed that DODGSON SCORE with respect to the parameter number of swaps (see also Section 3.2 for the corresponding definition) is unlikely to admit a polynomial-size kernel. This result is obtained by a polynomial parameter transformation from SMALL UNIVERSE HITTING SET which is known to be unlikely to have a polynomial-size problem kernel [DLS09].

Partial and Turing Kernels. There are cases where it seems hard to bound the whole size of the instance by a polynomial function in the parameter, but it is possible to bound only one dimension²⁰ of the input by such function. Here, the concept of *partial kernelization* [BGKN11] comes into play. For instance, for KEMENY SCORE one has a partial kernel with respect to the average KT-distance d_a , that is, one can construct equivalent instances with at most $16/3 \cdot d_a$ candidates, but the number of votes is unbounded [BBN10].

Furthermore, it could also be possible that one cannot find a problem kernel, but one is able to compute polynomially many instances whose sizes are bounded by some function in the parameter and the original instance is a yes-instance if

²⁰ In case of voting, for example the number of candidates or the number of votes are two natural dimensions.

and only if one of the new instances is a yes-instance. This leads to the concept of *Turing kernelization*, or to be more specific, disjunction truth-table kernelization [FFL⁺09, Lok09]. We are not aware of Turing kernelization results in voting.

Both, partial kernels and Turing kernels provide (similarly to the classical kernel concept) the possibility of obtaining fixed-parameter algorithms.

6.3 Dynamic Programming

The key idea of dynamic programming is to solve a problem by solving subproblems and to combine overlapping solutions to find an overall solution. Dynamic programming tries to avoid multiple computation of the same subsolution by storing it in a so-called *dynamic programming table*. It is a standard technique in mathematics and computer science and in the design of fixed-parameter algorithms [Nie06]. Often leading to very efficient algorithms, a typical bottleneck of dynamic programming is its memory consumption which may also be exponential in the parameter.

For instance, with dynamic programming one can solve KEMENY SCORE in $O^*(2^m)$ time (see Section 4.1). However, also the space requirement is $O^*(2^m)$.

A further example is DODGSON SCORE. Using dynamic programming it can be solved in $O(2^k \cdot nk \cdot nm)$ time [BGN10], where k denotes the number of swaps.

6.4 Integer Linear Programming

As one of the most popular techniques for problem solving, (integer) linear programming²¹ is also useful for classification and algorithm design in the context of parameterized algorithmics [Nie06]. A famous result of Lenstra [Len83] implies that a problem is fixed-parameter tractable when it can be solved by an integer linear program where the number of variables is upper-bounded by a function solely depending on the parameter.

Bartholdi et al. [BTT89b] developed an integer linear program to solve DODGSON SCORE and gave a running time bound based on Lenstra's result. They did not explicitly state this, but this shows fixed-parameter tractability for DODGSON SCORE with respect to the parameter number m of candidates. The corresponding integer linear program is shown in Figure 3. Note that it computes the Dodgson score of a specific candidate.

Although solvability by an integer linear program with a bounded number of variables implies fixed-parameter tractability, there is by far no guarantee for practically efficient algorithms. Indeed, due to a huge exponential function in the number of variables being part of the running time bound, Lenstra's [Len83] result is basically for classification only.

There are several similar fixed-parameter tractability results with respect to the parameter number of candidates for control problems [FHHR09b], POSSIBLE WINNER [BHN09], and SWAP BRIBERY [DS12] for various voting protocols.

²¹ See, for example, Matoušek and Gärtner [MG06] for a general introduction to linear programming.

$$\begin{aligned}
 & \min \sum_{i,j} j \cdot x_{i,j} \text{ subject to} \\
 & \forall i \in \tilde{V} : \sum_j x_{i,j} = N_i \\
 & \forall y \in C : \sum_{i,j} e_{i,j,y} \cdot x_{i,j} \geq d_y \\
 & x_{i,j} \geq 0
 \end{aligned}$$

Fig. 3. Integer linear program determining the Dodgson score of candidate c . Here, C denotes the set of candidates, \tilde{V} denotes the set of ranking types (that is, the set of votes where identical votes appear only once), N_i denotes the number of votes of type i , $x_{i,j}$ denotes the number of votes with rankings of type i for which candidate c will be moved upwards by j positions, $e_{i,j,y}$ is 1 if the result of moving candidate c by j positions upward in a ranking of type i is that c gains an additional vote against candidate y , and 0 otherwise. Furthermore, d_y is the deficit of c with respect to candidate y , that is, the minimum number of votes that c must gain against y to defeat her in a pairwise comparison. If c already defeats y , then $d_y = 0$. For more details see Bartholdi et al. [BTT89b]. Altogether, the integer linear program contains at most $m \cdot m!$ variables $x_{i,j}$ and at most $m! + m$ non-trivial constraints, where m denotes the number of candidates.

Furthermore, Dorn and Schlotter [DS12] convey without details that their result concerning SWAP BRIBERY can be transferred to problems like OPTIMAL LOBBYING and MANIPULATION under some specific voting protocols as well.

6.5 Color-Coding

Alon et al. [AYZ95] introduced *color-coding* as a randomized algorithm for solving some types of graph problems. Recently, Dorn and Schlotter [DS12] used it to show the fixed-parameter tractability of k -APPROVAL SWAP BRIBERY with respect to the number of votes for constant k (see Section 5.4). Here we sketch the idea behind their randomized fixed-parameter algorithm and how it employs color-coding.

Let I be an instance of k -APPROVAL SWAP BRIBERY consisting of an election (C, V) with $|C| = m$ and $|V| = n$, a distinguished candidate $d \in C$, a budget $\beta \in \mathbb{N}$, and a cost function $c : V \times C \times C \rightarrow \mathbb{N}$. Instance I is a yes-instance if and only if there is a sequence Γ of swaps with $\sum_{s \in \Gamma} c(s) \leq \beta$, and d wins after the swaps in Γ have been performed. We denote the votes after having performed Γ as $V^\Gamma = \bigcup_{v \in V} v^\Gamma$.

A candidate is *relevant* with respect to Γ if it receives at least one point in V^Γ . Let $C^{\text{rel}}(\Gamma)$ be the set of candidates relevant with respect to Γ . Since each of the first k candidates of a vote receives one point according to k -Approval, and since there are at most nk relevant candidates, we can identify each vote through a *vote pattern* which is a size- k subset of $\{1, \dots, nk\}$. It should represent the set of the first k candidates. An *election pattern* $P = (p_1, \dots, p_n)$ is an n -tuple of vote

patterns. There are $\binom{nk}{k}^n < (nk)^{nk}$ such election patterns. If the distinguished candidate d wins the bribed election, then we can assume that $d \in C^{\text{rel}}(\Gamma)$. We also require that d represents the number 1. Thus, an election pattern P is called *successful* if 1 appears at least as frequently as any other number between 2 and nk in P .

The basic idea of the algorithm is as follows: For each successful election pattern $P = (p_1, \dots, p_n)$, we color each candidate (except for candidate d) randomly with one of the colors of $\bigcup_{p \in P} p \setminus \{1\}$; d has color 1. If I is a yes-instance, then with probability of at least $(nk - 1)^{1-nk}$ we can find in $(nk)^{nk} \cdot \mathcal{O}(m^{k+1})$ randomized time a sequence Γ of swaps²² with the following properties: each relevant candidate in C^{rel} has a different color (while p has color 1), the colors of relevant candidates of vote $v_i^\Gamma \in V^\Gamma$ form the vote pattern $p_i \in P$, and the budget is not exceeded. Trying all possible successful election patterns, the algorithm takes a total of $(nk)^{2nk} \mathcal{O}(m^{k+1})$ randomized time. Note that using nk -perfect hash functions [AYZ95], one gains a deterministic fixed-parameter algorithm with respect to the parameter n for constant k .

6.6 Parameterized Intractability

There are several voting problems where computational intractability can be desirable for a protocol. Intractability in terms of parameterized complexity means $W[t]$ -hardness for some integer $t \geq 1$ (see Section 2). Without going into the details of the theory, the main message is that $W[t]$ -hard problems are not fixed-parameter tractable under several widely believed complexity assumptions (including the Exponential Time Hypothesis [IPZ01]). We present a simple parameterized reduction showing $W[2]$ -hardness in what follows; refer to the textbooks [DF99, FG06, Nie06] for general accounts.

To the best of our knowledge, the first W -hardness result result for a computational social choice problem is due to Christian et al. [CFRS07]. They considered the problem OPTIMAL LOBBYING as defined in Section 5.6.

Parameterized Reduction for Optimal Lobbying. The idea behind the proof of parameterized intractability for OPTIMAL LOBBYING is to describe a parameterized reduction (see Section 2) from the $W[2]$ -complete problem DOMINATING SET to OPTIMAL LOBBYING [CFRS07] (see Section 5.6). DOMINATING SET asks, given an undirected graph $G = (V, E)$, whether there exists a size- k subset of vertices $V' \subseteq V$ such that every vertex is either from V' or has a neighbor in V' . Such a vertex subset is called *dominating set*.

The construction of the OPTIMAL LOBBYING instance works as follows.²³ The matrix W is an extension of the adjacency matrix of G . First take the adjacency matrix of G and add one additional *selection column* filled with 1s. Then, add

²² See Dorn and Schlotter [DS12] for the detailed algorithm to find the swap sequence Γ when a successful election pattern and a feasible coloring are given.

²³ Note that the original construction works with interchanged roles for 1 and 0. This slight modification allows for a compact way of only presenting the idea.

$|V| - 2k + 1$ *dummy rows* filled with 0s. We call the original $|V|$ rows *vertex rows* and the original $|V|$ columns *vertex columns*. Finally, for each vertex column i , flip $|V| - k - N[i] + 1$ entries in the dummy rows from 0 to 1, where $N[i]$ denotes the number of neighbors of the vertex corresponding to column i . The target vector x (see Section 5.6) has a 0 in each of the $|V| + 1$ positions.

Now, we have the following situation. First, consider the selection column. The number of 1s exceeds the number of 0s by $2k - 1$, that is, k of the graph rows must be chosen in the solution. Now, consider the vertex columns. For each column the number of 1s exceeds the number of 0s only by 1. Hence, there is a majority for 0s if and only if the chosen vertex rows correspond to vertices forming a dominating set. In analogy, every dominating set implies such a solution.

Roughly speaking, this means that OPTIMAL LOBBYING remains intractable even if the number of voters to influence is small.

7 Discussion and Future Challenges

So far, the consideration of problems from algorithmic graph theory prevails in parameterized complexity studies. The impact of parameterized complexity analysis, however, strongly hinges on its high potential to explain, to predict, and to engineer computational complexity. The “computational complexity landscape” of problems arising in real-world applications needs a more fine-grained consideration than classical (one-dimensional) complexity analysis delivers. Thus, in 2008, two issues of *The Computer Journal* (Volume 51, Numbers 1 and 3 edited by Rod G. Downey, Michael R. Fellows, and Michael A. Langston) cover applications of parameterized complexity analysis in bioinformatics, computational geometry, artificial intelligence, constraint satisfaction, data bases, and cognitive modelling. Clearly, this list is far from being complete and deserves further additions. With this survey, we try to overview and promote the research on parameterized (and multivariate) complexity of voting problems, a subfield of the strongly growing area of computational social choice. Indeed, voting problems seem to be a particularly fruitful ground of (future) parameterized complexity analysis for at least three reasons:

- many NP-hard voting problems have simple and clear combinatorial definitions;
- many voting problems carry very natural structural parameters such as the number of candidates or the number of votes, with application scenarios where these parameter values are anticipated to be small;
- it is very natural and sometimes forcing to search for *exact* solutions.

The parameterized complexity analysis of voting problems leaves numerous challenges for future research. Some of these have been indicated in the preceding sections. Moreover, there are many NP-hard voting problems that have not yet been studied from a parameterized complexity perspective.

We conclude with a few more specific research questions and directions concerning the parameterized computational complexity of NP-hard voting problems (refer to a recent PhD thesis [Bet10a] for additional material).

- A central parameter in voting problems is the number of candidates (equivalently, alternatives). There is a number of fixed-parameter tractability results for this parameter [Bet10a, BHN09, DS12, EFS10b, FHHR09b] relying on integer linear programming and exploiting Lenstra’s result [Len83] for a fixed number of variables. It would be highly desirable to replace these results by direct combinatorial algorithms with more efficient running times.
- There are numerous results in the theory of voting [ASS02, ASS10] providing structural properties of specific voting systems. These might be exploited for spotting interesting parameters in voting problems. For instance, Elkind et al. [EFS10b] explored and exploited “distance rationalizability” to show fixed-parameter tractability results. Pini et al. [PRVW11] used “independence of irrelevant alternatives” to even gain polynomial-time solvability for a restricted POSSIBLE WINNER voting problem.
- From an algorithmic point of view, the established parameterized technique iterative compression [RSV04, GMN09] seems widely unexplored. Moreover, there are only few kernelization results in voting (see Section 6.2). Notably, it seems difficult to come up with kernelizations for the parameter “number of votes”. Hence, this calls for combined parameters in the spirit of multivariate algorithmics [Fel09, Nie10] or the development of partial kernelizations [BBN10, BGKN11] where only one input dimension is reduced.
- Many fixed-parameter tractability results in voting (as in algorithmic graph theory) are of theoretical nature only. It remains a general task to improve the efficiency of these results and to finally arrive at implementations and experiments in the spirit of algorithm engineering. For instance, algorithm engineering for computing Kemeny scores revealed that data reduction (based on partial kernelization results) combined with (integer) linear program solvers leads to practically relevant results [BBN10].
- Voting is an ideal playground for multivariate algorithmics [Fel09, Nie10]. In particular, for identifying (structural) parameters to exploit, it seems worthwhile to explore many of the NP-hardness proofs for voting problems. For instance, a proof showing NP-hardness for KEMENY SCORE with only four votes [DKNS01a, DKNS01b] reveals that in order to work, it requires a high average KT-distance between the votes, making this a plausible parameter. Hence, “deconstructing intractability” [KNU11, Nie10] appears particularly beneficial in case of voting problems.
- Voting provides numerous challenging combinatorial problems which are not about graphs.²⁴ However, directed graph problems pop up in many voting problems. For instance, there are close connections (also employed for proving NP-hardness results) between voting and problems on tournaments [BBS11, KS10, Woe03] or control in voting and vertex deletion problems on directed graphs [BBNU12, BU09, FHHR09b].

²⁴ In August 2011, Michael R. Fellows and Frances A. Rosamond organized at Charles Darwin University, Australia, the *Workshop-Parameterized Complexity: Not About Graphs (NAG)* in order to stimulate more parameterized complexity research beyond graph problems.

Voting problems are highly attractive from a parameterized complexity analysis perspective; this survey hopefully helps to attract more parameterized research in this fruitful and important area. Be invited!

Acknowledgements. We are grateful to Britta Dorn, Piotr Faliszewski, Jiong Guo, Matthias Mnich, Jörg Rothe, Ildikó Schlotter, and an anonymous referee for their numerous insightful remarks and their constructive advice.

References

- [AB09] Arora, S., Barak, B.: Computational Complexity: A Modern Approach. Cambridge University Press (2009)
- [Ail10] Ailon, N.: Aggregation of Partial Rankings, p -Ratings, and Top- m Lists. *Algorithmica* 57(2), 284–300 (2010)
- [ALS09] Alon, N., Lokshtanov, D., Saurabh, S.: Fast FAST. In: Albers, S., Marchetti-Spaccamela, A., Matias, Y., Nikolettseas, S., Thomas, W. (eds.) ICALP 2009, Part I. LNCS, vol. 5555, pp. 49–58. Springer, Heidelberg (2009)
- [ASS02] Arrow, K.J., Sen, A.K., Suzumura, K. (eds.): Handbook of Social Choice and Welfare, vol. 1. North-Holland (2002)
- [ASS10] Arrow, K.J., Sen, A.K., Suzumura, K. (eds.): Handbook of Social Choice and Welfare, vol. 2. North-Holland (2010)
- [AYZ95] Alon, N., Yuster, R., Zwick, U.: Color-Coding. *Journal of the ACM* 42(4), 844–856 (1995)
- [Ban85] Banks, J.S.: Sophisticated Voting Outcomes and Agenda Control. *Social Choice and Welfare* 1(4), 295–306 (1985)
- [BBD09] Biedl, T.C., Brandenburg, F.-J., Deng, X.: On the Complexity of Crossings in Permutations. *Discrete Mathematics* 309(7), 1813–1823 (2009)
- [BBN10] Betzler, N., Bredereck, R., Niedermeier, R.: Partial Kernelization for Rank Aggregation: Theory and Experiments. In: Raman, V., Saurabh, S. (eds.) IPEC 2010. LNCS, vol. 6478, pp. 26–37. Springer, Heidelberg (2010)
- [BBNU11] Betzler, N., Bredereck, R., Niedermeier, R., Uhlmann, J.: On Making a Distinguished Vertex Minimum Degree by Vertex Deletion. In: Černá, I., Gyimóthy, T., Hromkovič, J., Jefferey, K., Královič, R., Vukolić, M., Wolf, S. (eds.) SOFSEM 2011. LNCS, vol. 6543, pp. 123–134. Springer, Heidelberg (2011)
- [BBNU12] Betzler, N., Bredereck, R., Niedermeier, R., Uhlmann, J.: On Bounded-Degree Vertex Deletion Parameterized by Treewidth. *Discrete Applied Mathematics* 160(1–2), 53–60 (2012)
- [BBS11] Brandt, F., Brill, M., Seedig, H.G.: On the Fixed-Parameter Tractability of Composition-Consistent Tournament Solutions. In: Proceedings of the 22nd International Joint Conference on Artificial Intelligence, pp. 85–90. AAAI Press (2011)
- [BCE12] Brandt, F., Conitzer, V., Endriss, U.: Computational Social Choice. In: Weiss, G. (ed.) Multiagent Systems. MIT Press (2012)
- [BD10] Betzler, N., Dorn, B.: Towards a Dichotomy of Finding Possible Winners in Elections Based on Scoring Rules. *Journal of Computer and System Sciences* 76(8), 812–836 (2010)

- [BDFH09] Bodlaender, H.L., Downey, R.G., Fellows, M.R., Hermelin, D.: On Problems Without Polynomial Kernels. *Journal of Computer and System Sciences* 75(8), 423–434 (2009)
- [BEH⁺10] Baumeister, D., Erdélyi, G., Hemaspaandra, E., Hemaspaandra, L.A., Rothe, J.: Computational Aspects of Approval Voting. In: Laslier, J.-F., Remzi Sanver, M. (eds.) *Handbook on Approval Voting*, ch. 10, pp. 199–251. Springer (2010)
- [Bet10a] Betzler, N.: A Multivariate Complexity Analysis of Voting Problems. PhD thesis, Friedrich-Schiller-Universität Jena (2010)
- [Bet10b] Betzler, N.: On Problem Kernels for Possible Winner Determination under the k -Approval Protocol. In: Hliněný, P., Kučera, A. (eds.) *MFCS 2010. LNCS*, vol. 6281, pp. 114–125. Springer, Heidelberg (2010)
- [BF02] Brams, S., Fishburn, P.C.: Voting Procedures. In: Arrow, K.J., Sen, A.K., Suzumura, K. (eds.) *Handbook of Social Choice and Welfare*, vol. 1, pp. 173–236. Elsevier (2002)
- [BFG⁺09] Betzler, N., Fellows, M.R., Guo, J., Niedermeier, R., Rosamond, F.A.: Fixed-Parameter Algorithms for Kemeny Rankings. *Theoretical Computer Science* 410, 4554–4570 (2009)
- [BGKN11] Betzler, N., Guo, J., Komusiewicz, C., Niedermeier, R.: Average Parameterization and Partial Kernelization for Computing Medians. *Journal of Computer and System Sciences* 77, 774–789 (2011)
- [BGN10] Betzler, N., Guo, J., Niedermeier, R.: Parameterized Computational Complexity of Dodgson and Young Elections. *Information and Computation* 208(2), 165–177 (2010)
- [BHN09] Betzler, N., Hemmann, S., Niedermeier, R.: A Multivariate Complexity Analysis of Determining Possible Winners Given Incomplete Votes. In: *Proceedings of the 21st International Joint Conference on Artificial Intelligence*, pp. 53–58 (2009)
- [BNW11] Betzler, N., Niedermeier, R., Woeginger, G.J.: Unweighted Coalitional Manipulation Under the Borda Rule is NP-hard. In: *Proceedings of 22nd International Joint Conference of Artificial Intelligence*, pp. 55–60 (2011)
- [BO91] Bartholdi III, J.J., Orlin, J.B.: Single Transferable Vote Resists Strategic Voting. *Social Choice and Welfare* 8, 341–354 (1991)
- [Bod09] Bodlaender, H.L.: Kernelization: New Upper and Lower Bound Techniques. In: Chen, J., Fomin, F.V. (eds.) *IWPEC 2009. LNCS*, vol. 5917, pp. 17–37. Springer, Heidelberg (2009)
- [BS09] Brams, S., Remzi Sanver, M.: Voting Systems that Combine Approval and Preference. In: Brams, S., Gehrlein, W.V., Roberts, F.S. (eds.) *The Mathematics of Preference, Choice, and Order: Essays in Honor of Peter C. Fishburn*, pp. 215–237. Springer (2009)
- [BSU11] Betzler, N., Slinko, A., Uhlmann, J.: On the Computation of Fully Proportional Representation (2011) (available at Social Science Research Network)
- [BTT89a] Bartholdi III, J.J., Tovey, C.A., Trick, M.A.: The Computational Difficulty of Manipulating an Election. *Social Choice and Welfare* 6(3), 227–241 (1989)
- [BTT89b] Bartholdi III, J.J., Tovey, C.A., Trick, M.A.: Voting Schemes for Which It Can Be Difficult to Tell Who Won the Election. *Social Choice and Welfare* 6(2), 157–165 (1989)
- [BTT92] Bartholdi III, J.J., Tovey, C.A., Trick, M.A.: How Hard Is It to Control an Election? *Mathematical and Computer Modeling* 16(8-9), 27–40 (1992)

- [BTY11] Bodlaender, H.L., Thomassé, S., Yeo, A.: Kernel Bounds for Disjoint Cycles and Disjoint Paths. *Theoretical Computer Science* 412(35), 4570–4578 (2011)
- [BU09] Betzler, N., Uhlmann, J.: Parameterized Complexity of Candidate Control in Elections and Related Digraph Problems. *Theoretical Computer Science* 410(52), 5425–5442 (2009)
- [CC83] Chamberlin, J.R., Courant, P.N.: Representative Deliberations and Representative Decisions: Proportional Representation and the Borda Rule. *American Political Science Review* 77(3), 718–733 (1983)
- [CCDF97] Cai, L., Chen, J., Downey, R.G., Fellows, M.R.: Advice Classes of Parameterized Tractability. *Annals of Pure and Applied Logic* 84, 119–138 (1997)
- [CELM07] Chevalyere, Y., Endriss, U., Lang, J., Maudet, N.: A Short Introduction to Computational Social Choice. In: van Leeuwen, J., Italiano, G.F., van der Hoek, W., Meinel, C., Sack, H., Plášil, F. (eds.) *SOFSEM 2007. LNCS*, vol. 4362, pp. 51–69. Springer, Heidelberg (2007)
- [CFRS07] Christian, R., Fellows, M., Rosamond, F., Slinko, A.: On Complexity of Lobbying in Multiple Referenda. *Review of Economic Design* 11(3), 217–224 (2007)
- [CH00] Charon, I., Hudry, O.: Slater Orders and Hamiltonian Paths of Tournaments. *Electronic Notes in Discrete Mathematics* 5, 60–63 (2000)
- [CLM⁺11] Chevalyere, Y., Lang, J., Maudet, N., Monnot, J., Xia, L.: New Candidates Welcome! Possible Winners with Respect to the Addition of New Candidates. In: *CoRR*, abs/1111.3690 (2011)
- [Con06] Conitzer, V.: Computing Slater Rankings Using Similarities among Candidates. In: *Proceedings of the 21st AAAI Conference on Artificial Intelligence*, pp. 613–619. AAAI Press (2006)
- [Con10] Conitzer, V.: Making Decisions Based on the Preferences of Multiple Agents. *Communications of the ACM* 53, 84–94 (2010)
- [Cop51] Copeland, A.H.: A ‘Resonable’ Social Welfare Function. Mimeographed (University of Michigan Seminar on Application of Mathematics in Social Science) (1951)
- [CRX09] Conitzer, V., Rognlie, M., Xia, L.: Preference Functions That Score Rankings and Maximum Likelihood Estimation. In: *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pp. 109–115 (2009)
- [CSL07] Conitzer, V., Sandholm, T., Lang, J.: When Are Elections with Few Candidates Hard to Manipulate? *Journal of the ACM* 54, 1–33 (2007)
- [dB81] de Borda, J.-C.: *Mémoire sur les élections au scrutin*. Histoire de l’Académie Royale des Sciences (1781)
- [dC85] Caritat, M.J.A.N., de Condorcet: *Essai sur l’application de l’analyse à la probabilité des décisions rendues à la pluralité des voix*. L’Imprimerie Royale, Paris (1785)
- [DF99] Downey, R.G., Fellows, M.R.: *Parameterized Complexity*. Springer (1999)
- [DKNS01a] Dwork, C., Kumar, R., Naor, M., Sivakumar, D.: Rank Aggregation Methods for the Web. In: *Proceedings of the 10th International Conference on World Wide Web*, pp. 613–622. ACM (2001)
- [DKNS01b] Dwork, C., Kumar, R., Naor, M., Sivakumar, D.: Rank Aggregation Revisited (2001) (manuscript)
- [DKNW11] Davies, J., Katsirelos, G., Narodytska, N., Walsh, T.: Complexity of and Algorithms for Borda Manipulation. In: *Proceedings of the 25th AAAI Conference on Artificial Intelligence*, pp. 657–662. AAAI Press (2011)

- [DLS09] Dom, M., Lokshtanov, D., Saurabh, S.: Incompressibility through Colors and IDs. In: Albers, S., Marchetti-Spaccamela, A., Matias, Y., Nikolettseas, S., Thomas, W. (eds.) ICALP 2009, Part I. LNCS, vol. 5555, pp. 378–389. Springer, Heidelberg (2009)
- [Dod76] Dodgson, C.: A Method of Taking Votes on More Than Two Issues. Pamphlet printed by the Clarendon Press, Oxford, and headed (1876) (not yet published)
- [DS12] Dorn, B., Schlotter, I.: Multivariate Complexity Analysis of Swap Bribery. *Algorithmica* (2012) (available electronically)
- [DT11] Downey, R.G., Thilikos, D.M.: Confronting Intractability via Parameters. *Computer Science Review* 5(4), 279–317 (2011)
- [EF10a] Elkind, E., Faliszewski, P.: Approximation Algorithms for Campaign Management. In: Saberi, A. (ed.) WINE 2010. LNCS, vol. 6484, pp. 473–482. Springer, Heidelberg (2010)
- [EF10b] Erdélyi, G., Fellows, M.R.: Parameterized Control Complexity in Bucklin Voting and in Fallback Voting. In: Proceedings of the 3rd International Workshop on Computational Social Choice, pp. 163–174 (2010)
- [EFG⁺09] Erdélyi, G., Fernau, H., Goldsmith, J., Mattei, N., Raible, D., Rothe, J.: The Complexity of Probabilistic Lobbying. In: Rossi, F., Tsoukias, A. (eds.) ADT 2009. LNCS, vol. 5783, pp. 86–97. Springer, Heidelberg (2009)
- [EFPR11] Erdélyi, G., Fellows, M.R., Piras, L., Rothe, J.: Control Complexity in Bucklin and Fallback Voting. Technical report, arXiv:1103.2230 (2011)
- [EFS09] Elkind, E., Faliszewski, P., Slinko, A.: Swap Bribery. In: Mavronicolas, M., Papadopoulos, V.G. (eds.) SAGT 2009. LNCS, vol. 5814, pp. 299–310. Springer, Heidelberg (2009)
- [EFS10a] Elkind, E., Faliszewski, P., Slinko, A.: Cloning in Elections. In: Proceedings of the 24th AAAI Conference on Artificial Intelligence, pp. 768–773. AAAI Press (2010)
- [EFS10b] Elkind, E., Faliszewski, P., Slinko, A.: On the Role of Distances in Defining Voting Rules. In: Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems, pp. 375–382 (2010)
- [ENR09] Erdélyi, G., Nowak, M., Rothe, J.: Sincere-Strategy Preference-Based Approval Voting Fully Resists Constructive Control and Broadly Resists Destructive Control. *Mathematical Logic Quarterly* 55, 425–443 (2009)
- [EPR10] Erdélyi, G., Piras, L., Rothe, J.: Control Complexity in Fallback Voting. Technical report, arXiv:1004.3398v1 (2010)
- [ER91] Ephrati, E., Rosenschein, J.S.: The Clarke Tax as a Consensus Mechanism Among Automated Agents. In: Proceedings of the 9th AAAI Conference on Artificial Intelligence, pp. 173–178. AAAI Press (1991)
- [ER97] Ephrati, E., Rosenschein, J.S.: A Heuristic Technique for Multi-Agent Planning. *Annals of Mathematics and Artificial Intelligence* 20(1–4), 13–67 (1997)
- [ER10] Erdélyi, G., Rothe, J.: Control Complexity in Fallback Voting. In: Proceedings of Computing: the 16th Australasian Theory Symposium. Australian Computer Society Conferences in Research and Practice in Information Technology Series, pp. 39–48 (2010)
- [Fal08] Faliszewski, P.: Nonuniform Bribery. In: Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems. International Foundation for Autonomous Agents and Multiagent Systems, pp. 1569–1572 (2008)

- [Fel09] Fellows, M.: Towards Fully Multivariate Algorithmics: Some New Results and Directions in Parameter Ecology. In: Fiala, J., Kratochvíl, J., Miller, M. (eds.) IWOCA 2009. LNCS, vol. 5874, pp. 2–10. Springer, Heidelberg (2009)
- [FFL⁺09] Fernau, H., Fomin, F.V., Lokshtanov, D., Raible, D., Saurabh, S., Villanger, Y.: Kernel(s) for Problems with No Kernel: On Out-Trees with Many Leaves. In: Proceedings of the 26th International Symposium on Theoretical Aspects of Computer Science. LIPIcs, vol. 3, pp. 421–432. Schloss Dagstuhl (2009)
- [FFL⁺10] Fernau, H., Fomin, F.V., Lokshtanov, D., Mnich, M., Philip, G., Saurabh, S.: Ranking and Drawing in Subexponential Time. In: Iliopoulos, C.S., Smyth, W.F. (eds.) IWOCA 2010. LNCS, vol. 6460, pp. 337–348. Springer, Heidelberg (2011)
- [FG06] Flum, J., Grohe, M.: Parameterized Complexity Theory. Springer (2006)
- [FHH09] Faliszewski, P., Hemaspaandra, E., Hemaspaandra, L.A.: How Hard Is Bribery in Elections? *Journal of Artificial Intelligence Research* 35, 485–532 (2009)
- [FHH10] Faliszewski, P., Hemaspaandra, E., Hemaspaandra, L.A.: Using Complexity to Protect Elections. *Communications of the ACM* 53(11), 74–82 (2010)
- [FHH11] Faliszewski, P., Hemaspaandra, E., Hemaspaandra, L.A.: Multimode Control Attacks on Elections. *Journal of Artificial Intelligence Research* 40, 305–351 (2011)
- [FHHR09a] Faliszewski, P., Hemaspaandra, E., Hemaspaandra, L.A., Rothe, J.: A Richer Understanding of the Complexity of Election Systems. In: *Fundamental Problems in Computing: Essays in Honor of Professor Daniel J. Rosenkrantz*, pp. 375–406 (2009)
- [FHHR09b] Faliszewski, P., Hemaspaandra, E., Hemaspaandra, L.A., Rothe, J.: Llull and Copeland Voting Computationally Resist Bribery and Constructive Control. *Journal of Artificial Intelligence Research* 35, 275–341 (2009)
- [Fis77] Fishburn, P.C.: Condorcet Social Choice Functions. *SIAM Journal on Applied Mathematics* 33(3), 469–489 (1977)
- [FJL⁺10] Fellows, M.R., Jansen, B., Lokshtanov, D., Rosamond, F.A., Saurabh, S.: Determining the Winner of a Dodgson Election is Hard. In: *Proceedings of the 29th Conference on Foundations of Software Technology and Theoretical Computer Science*, pp. 459–469. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2010)
- [FKS03] Fagin, R., Kumar, R., Sivakumar, D.: Efficient Similarity Search and Classification via Rank Aggregation. In: *Proceedings of the 22nd ACM SIGMOD International Conference on Management of Data*, pp. 301–312. ACM (2003)
- [FS11] Fortnow, L., Santhanam, R.: Infeasibility of Instance Compression and Succinct PCPs for NP. *Journal of Computer and System Sciences* 77(1), 91–106 (2011)
- [Gae09] Gaertner, W.: *A Primer in Social Choice Theory—LSE Perspectives in Economic Analysis*, revised edition. Oxford University Press (2009)
- [Gib73] Gibbard, A.: Manipulation of Voting Schemes: A General Result. *Econometrica* 41(4), 587–601 (1973)
- [GMN09] Guo, J., Moser, H., Niedermeier, R.: Iterative Compression for Exactly Solving NP-Hard Minimization Problems. In: Lerner, J., Wagner, D., Zweig, K.A. (eds.) *Algorithmics of Large and Complex Networks*. LNCS, vol. 5515, pp. 65–80. Springer, Heidelberg (2009)

- [GN07] Guo, J., Niedermeier, R.: Invitation to Data Reduction and Problem Kernelization. *ACM SIGACT News* 38(1), 31–45 (2007)
- [Goo54] Goodman, L.A.: On Methods of Amalgamation. In: Thrall, R.M., Coombs, C.H., Davis, R.L. (eds.) *Decision Processes*, pp. 39–48. John Wiley and Sons, Inc. (1954)
- [HHR97] Hemaspaandra, E., Hemaspaandra, L.A., Rothe, J.: Exact Analysis of Dodgson Elections: Lewis Carroll’s 1876 Voting System is Complete for Parallel Access to NP. *Journal of the ACM* 44(6), 806–825 (1997)
- [HHR07] Hemaspaandra, E., Hemaspaandra, L.A., Rothe, J.: Anyone but Him: The Complexity of Precluding an Alternative. *Artificial Intelligence* 171(5-6), 255–285 (2007)
- [HHR09] Hemaspaandra, E., Hemaspaandra, L.A., Rothe, J.: Hybrid Elections Broaden Complexity-Theoretic Resistance to Control. *Mathematical Logic Quarterly* 55(4), 397–424 (2009)
- [HSV05] Hemaspaandra, E., Spakowski, H., Vogel, J.: The Complexity of Kemeny Elections. *Theoretical Computer Science* 349(3), 382–391 (2005)
- [Hud04] Hudry, O.: A Note On “Banks Winners in Tournaments Are Difficult to Recognize” by G. J. Woeginger. *Social Choice and Welfare* 23(1), 113–114 (2004)
- [IP01] Impagliazzo, R., Paturi, R.: On the Complexity of k -SAT. *Journal of Computer and System Sciences* 62, 367–375 (2001)
- [IPZ01] Impagliazzo, R., Paturi, R., Zane, F.: Which Problems Have Strongly Exponential Complexity? *Journal of Computer and System Sciences* 63(4), 512–530 (2001)
- [JSA08] Jackson, B.N., Schnable, P.S., Aluru, S.: Consensus Genetic Maps as Median Orders from Inconsistent Sources. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 5(2), 161–171 (2008)
- [Kem59] Kemeny, J.G.: *Mathematics Without Numbers*. Daedalus 88, 571–591 (1959)
- [KL05] Konczak, K., Lang, J.: Voting Procedures with Incomplete Preferences. In: *Proceedings of IJCAI 2005 Multidisciplinary Workshop on Advances in Preference Handling*, pp. 124–129 (2005)
- [KNU11] Komusiewicz, C., Niedermeier, R., Uhlmann, J.: Deconstructing Intractability—A Multivariate Complexity Analysis of Interval Constrained Coloring. *Journal of Discrete Algorithms* 9, 137–151 (2011)
- [KS10] Karpinski, M., Schudy, W.: Faster Algorithms for Feedback Arc Set Tournament, Kemeny Rank Aggregation and Betweenness Tournament. In: Cheong, O., Chwa, K.-Y., Park, K. (eds.) *ISAAC 2010*. LNCS, vol. 6506, pp. 3–14. Springer, Heidelberg (2010)
- [KT06] Kleinberg, J., Tardos, É.: *Algorithm Design*. Addison-Wesley (2006)
- [LB11] Lu, T., Boutilier, C.: Budgeted Social Choice: From Consensus to Personalized Decision Making. In: *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, pp. 280–286 (2011)
- [Len83] Lenstra, H.W.: Integer Programming with a Fixed Number of Variables. *Mathematics of Operations Research* 8(4), 538–548 (1983)
- [Lev75] Levenglick, A.: Fair and Reasonable Election Systems. *Behavioral Science* 20(1), 34–46 (1975)
- [LFZL09] Liu, H., Feng, H., Zhu, D., Luan, J.: Parameterized Computational Complexity of Control Problems in Voting Systems. *Theoretical Computer Science* 410, 2746–2753 (2009)

- [Lok09] Lokshтанov, D.: *New Methods in Parameterized Algorithms and Complexity*. PhD thesis, University of Bergen (2009)
- [LR08] Lindner, C., Rothe, J.: *Fixed-Parameter Tractability and Parameterized Complexity Applied to Problems From Computational Social Choice*. In: *Supplement in the Mathematical Programming Glossary* (October 2008)
- [LZ10] Liu, H., Zhu, D.: *Parameterized Complexity of Control Problems in Maximin Election*. *Information Processing Letters* 110(10), 383–388 (2010)
- [MG06] Matoušek, J., Gärtner, B.: *Understanding and Using Linear Programming* (Universitext). Springer (2006)
- [Mon95] Monroe, B.L.: *Fully Proportional Representation*. *American Political Science Review* 89(4), 925–940 (1995)
- [Mou91] Moulin, H.: *Axioms of Cooperative Decision Making*. Cambridge University Press (1991)
- [MR99] Mahajan, M., Raman, V.: *Parameterizing Above Guaranteed Values: MaxSat and MaxCut*. *Journal of Algorithms* 31(2), 335–354 (1999)
- [MRS09] Mahajan, M., Raman, V., Sikdar, S.: *Parameterizing Above or Below Guaranteed Values*. *Journal of Computer and System Sciences* 75, 137–153 (2009)
- [MRS11] Misra, N., Raman, V., Saurabh, S.: *Lower Bounds on Kernelization*. *Discrete Optimization* 8(1), 110–128 (2011)
- [Nie06] Niedermeier, R.: *Invitation to Fixed-Parameter Algorithms*. Oxford University Press (February 2006)
- [Nie10] Niedermeier, R.: *Reflections on Multivariate Algorithmics and Problem Parameterization*. In: *Proceedings of the 27th International Symposium on Theoretical Aspects of Computer Science*. LIPIcs, vol. 5, pp. 17–32 (2010)
- [Nur87] Nurmi, H.: *Comparing Voting Systems*. Kluwer Academic Publishers (1987)
- [Pap94] Papadimitriou, C.H.: *Computational Complexity*. Addison-Wesley (1994)
- [PRVW11] Pini, M.S., Rossi, F., Brent Venable, K., Walsh, T.: *Incompleteness and Incomparability in Preference Aggregation: Complexity Results*. *Artificial Intelligence* 175, 1272–1289 (2011)
- [PRZ08] Procaccia, A.D., Rosenschein, J.S., Zohar, A.: *On the Complexity of Achieving Proportional Representation*. *Social Choice and Welfare* 30, 353–362 (2008)
- [RBLR11] Rothe, J., Baumeister, D., Lindner, C., Rothe, I.: *Einführung in Computational Social Choice: Individuelle Strategien und kollektive Entscheidungen beim Spielen*. Spektrum Akademischer Verlag, Wählen und Teilen (2011)
- [RS07] Raman, V., Saurabh, S.: *Improved Fixed Parameter Tractable Algorithms for Two “Edge” Problems: MAXCUT and MAXDAG*. *Information Processing Letters* 104(2), 65–72 (2007)
- [RSV03] Rothe, J., Spakowski, H., Vogel, J.: *Exact Complexity of the Winner Problem for Young Elections*. *Theory of Computing Systems* 36(4), 375–386 (2003)
- [RSV04] Reed, B.A., Smith, K., Vetta, A.: *Finding Odd Cycle Transversals*. *Operations Research Letters* 32, 299–301 (2004)
- [Sat75] Satterthwaite, M.A.: *Strategy-Proofness and Arrow’s Conditions: Existence and Correspondence Theorems for Voting Procedures and Social Welfare Functions*. *Journal of Economic Theory*, 187–217 (1975)
- [Scu07] Sculley, D.W.: *Rank Aggregation for Similar Items*. In: *Proceedings of the 7th SIAM International Conference on Data Mining*, pp. 587–592 (2007)

- [SEF11] Schlotter, I., Elkind, E., Faliszewski, P.: Campaign Management under Approval-Driven Voting Rules. In: Proceedings of the 25th AAAI Conference on Artificial Intelligence, pp. 726–731. AAAI Press (2011)
- [Sim09] Simjour, N.: Improved Parameterized Algorithms for the Kemeny Aggregation Problem. In: Chen, J., Fomin, F.V. (eds.) IWPEC 2009. LNCS, vol. 5917, pp. 312–323. Springer, Heidelberg (2009)
- [Sla61] Slater, P.: Inconsistencies in a Schedule of Paired Comparisons. *Biometrika* 48(3-4), 303–312 (1961)
- [Sni08] Sniedovich, M.: Wald’s Maximin Model: A Treasure in Disguise! *Journal of Risk Finance* 9(3), 287–291 (2008)
- [Tay05] Taylor, A.D.: *Social Choice and the Mathematics of Manipulation*. Cambridge University Press (2005)
- [Wal49] Wald, A.: Statistical Decision Functions. *The Annals of Mathematical Statistics* 20(2) (1949)
- [Wal07] Walsh, T.: Uncertainty in Preference Elicitation and Aggregation. In: Proceedings of the 22nd AAAI Conference on Artificial Intelligence, pp. 3–8. AAAI Press (2007)
- [Woe03] Woeginger, G.J.: Banks Winners in Tournaments are Difficult to Recognize. *Social Choice and Welfare* 20(3), 523–528 (2003)
- [XC11] Xia, L., Conitzer, V.: Determining Possible and Necessary Winners under Common Voting Rules Given Partial Orders. *Journal of Artificial Intelligence Research* 41, 25–67 (2011)
- [YL78] Young, H.P., Levenglick, A.: A Consistent Extension of Condorcet’s Election Principle. *SIAM Journal on Applied Mathematics* 35(2), 285–300 (1978)
- [You77] Young, H.P.: Extending Condorcet’s Rule. *Journal of Economic Theory* 16, 335–353 (1977)

A Parameterized Halting Problem

Yijia Chen¹ and Jörg Flum²

¹ Shanghai Jiaotong University, China
yijia.chen@cs.sjtu.edu.cn

² Albert-Ludwigs-Universität Freiburg, Germany
joerg.flum@math.uni-freiburg.de

Abstract. The parameterized problem p -HALT takes as input a nondeterministic Turing machine \mathbb{M} and a natural number n , the size of \mathbb{M} being the parameter. It asks whether every accepting run of \mathbb{M} on empty input tape takes more than n steps. This problem is in the class XP_{uni} , the class “uniform XP,” if there is an algorithm deciding it, which for fixed machine \mathbb{M} runs in time polynomial in n . It turns out that various open problems of different areas of theoretical computer science are related or even equivalent to p -HALT $\in \text{XP}_{\text{uni}}$. Thus this statement forms a bridge which allows to derive equivalences between statements of different areas (proof theory, complexity theory, descriptive complexity, ...) which at first glance seem to be unrelated. As our presentation shows, various of these equivalences may be obtained by the same method.

1 Introduction

Halting problems played a central role in *computability theory* right up from the beginning. In fact, in his seminal paper [33] Turing made precise the notion of algorithm by introducing the type of machine known as Turing machine and proved the first undecidability result: the undecidability of the halting problem for Turing machines:

Instance: A Turing machine \mathbb{M} .
Problem: Does \mathbb{M} accept the empty input tape?

In *complexity theory* Cook [13] and Levin [28] showed that the halting problem

NTM-HALT

Instance: A nondeterministic Turing machine \mathbb{M} and a string 1^n with $n \in \mathbb{N}$.

Problem: Does \mathbb{M} accept the empty input tape in $\leq n$ steps?

is NP-complete (under polynomial time reductions). Comparing this problem with other NP-complete problems, Downey and Fellows remark in [16, page 236]:

... the point is that a nondeterministic Turing machine is such an opaque and generic object that it simply does not seem reasonable that we should be able to decide in polynomial time whether a given Turing machine has some accepting path. The Cook-Levin theorem therefore gives powerful evidence that $P \neq NP$.

Later on Chandra, Kozen, and Stockmeyer [7] introduced the notion of alternating Turing machine and showed that all levels of the Polynomial Hierarchy have natural complete problems that are halting problems for alternating Turing machines.

Also in *parameterized complexity* most complexity classes of parameterized intractable problems considered so far contain a more or less natural complete halting problem (complete under fpt-reductions). Depending on the class the machines are deterministic, nondeterministic, or alternating and they are single or multitape machines. The reader will find the completeness results, for example, in the textbook [20] on parameterized complexity. In this introduction we mention some of these results, however, otherwise not used in this paper.

The classes $W[1]$, $W[2]$, and $W[P]$ are (among) the most important classes of parameterized intractable problems. In particular, for each of them there is a halting problem for nondeterministic Turing machines complete for it. In fact, the *short halting problem for nondeterministic single-tape Turing machines* p -SHORT-NSTM-HALT, the *short halting problem for nondeterministic (multitape) Turing machines* p -SHORT-NTM-HALT, and the *bounded halting problem for nondeterministic Turing machines* p -BOUNDED-NTM-HALT are complete for $W[1]$, $W[2]$, and $W[P]$, respectively, where:

p -SHORT-NSTM-HALT

Instance: A nondeterministic single tape Turing machine \mathbb{M}
and $k \in \mathbb{N}$.

Parameter: k .

Problem: Does \mathbb{M} accept the empty input tape in $\leq k$ steps?

p -SHORT-NTM-HALT

Instance: A nondeterministic multitape Turing machine \mathbb{M}
and $k \in \mathbb{N}$.

Parameter: k .

Problem: Does \mathbb{M} accept the empty input tape in $\leq k$ steps?

p -BOUNDED-NTM-HALT

Instance: A nondeterministic Turing machine \mathbb{M} , $k \in \mathbb{N}$,
and 1^n with $n \in \mathbb{N}$.

Parameter: k .

Problem: Does \mathbb{M} accept the empty input tape in $\leq n$ steps
and using at most k nondeterministic steps?

These completeness results for $W[1]$, $W[2]$, and $W[P]$ are due to Cai et al. [4], Cesati et al. [6], and Cesati [5], respectively. Concerning the $W[1]$ -completeness of p -SHORT-NSTM-HALT, Downey and Fellows write in [16, page 236]:

*It seems to us that if one accepts the philosophical argument that **TURING MACHINE ACCEPTANCE** [that is, NTM-HALT] is intractable, then the same reasoning would suggest that **SHORT TURING MACHINE ACCEPTANCE** [that is, p -SHORT-NSTM-HALT] is fixed parameter intractable.*

In the problem p -BOUNDED-NTM-HALT the parameter bounds the number of nondeterministic steps that are allowed in a run of length n , so small parameter means limited nondeterminism. However in the first two parameterized halting problems the parameter is the total number of steps considered in the given instance. Having in mind the original investigations which led to the introduction of the halting problem for (deterministic) Turing machines one hardly would argue that the parameter is small compared with the total size of an instance. In this context it is more natural to expect that the size of the Turing machine is small compared with the number of steps considered in a run of the machine on empty input tape.

If in p -SHORT-NSTM-HALT we replace the nondeterministic machines by alternating ones with the appropriate number of alternations we obtain complete parameterized halting problems for the different levels of the so-called A-hierarchy [19]. Finally, in this short report on known results concerning halting problems in parameterized complexity, let us mention at least one parameterized class with a halting problem for *deterministic* machines as complete problem. The problem

p -EXP-DTM-HALT

Instance: A Turing machine M , $k \in \mathbb{N}$ and 1^n with $n \in \mathbb{N}$.

Parameter: k .

Problem: Does M accept the empty input tape in at most n^k steps?

is complete for the class XP.

We already mentioned that in halting problems the size of the machine is a reasonable parameter (reasonable in the sense of parameterized complexity). In this paper we consider the parameterized halting problem:

p -HALT

Instance: A nondeterministic Turing machine M and 1^n with $n \in \mathbb{N}$.

Parameter: $|M|$, the size of M .

Problem: Does every accepting run of M on empty input tape take more than n steps?

We introduced this parameterization of the halting problem in [9]; independently, it was introduced by Aumann and Domb [1]. Later on the complexity

of this problem has also been studied by Monroe [30]. Note that the classical problem HALT underlying p -HALT essentially is the complement of the problem NTM-HALT considered above. In fact, $\langle \mathbb{M}, 1^n \rangle \in p$ -HALT means that there is no accepting run of \mathbb{M} on empty input tape at all or if there is one, then all such runs take more than n steps.

We have seen above that (apparently) it makes a difference whether we consider single tape or multitape machines. Moreover, if we restrict the inputs of p -SHORT-NSTM-HALT to nondeterministic Turing machines with the cardinality of its alphabet bounded by some constant, then the problem becomes fixed-parameter tractable. As in p -HALT the size of the machine is the parameter, its complexity is robust against such changes (fixed alphabet versus arbitrary alphabet, single tape versus multitape, binary branching versus finite branching, ...); in fact, one easily verifies that any two such variants are fpt-equivalent.

It is easily seen that HALT (the classical problem underlying p -HALT) is coNP-complete. The algorithm that for every instance $\langle \mathbb{M}, 1^n \rangle$ of p -HALT systematically checks all possible runs of length $\leq n$ of \mathbb{M} on empty input tape takes $|\mathbb{M}|^n$ steps approximately. The “small” parameter $|\mathbb{M}|$ is the base of the term $|\mathbb{M}|^n$ and the “big” n its exponent. The question arises whether we can reverse the roles of $|\mathbb{M}|$ and n , more precisely, whether there is an algorithm solving $\langle \mathbb{M}, 1^n \rangle \in p$ -HALT in time $n^{f(|\mathbb{M}|)}$ for some function $f : \mathbb{N} \rightarrow \mathbb{N}$, that is, whether

$$p\text{-HALT} \in \text{XP}_{\text{uni}}.$$

This problem is widely open. We conjecture that p -HALT $\notin \text{XP}_{\text{uni}}$; in fact, encouraged by the remarks of Downey and Fellows mentioned above, we are tempted to add:

The point is that a nondeterministic Turing machine is such an opaque and generic object that it simply does not seem reasonable that we should be able to decide whether every accepting run on empty input tape of a given nondeterministic Turing machine \mathbb{M} takes more than n steps in time $n^{f(|\mathbb{M}|)}$ for some function f .

In this paper first we report on what is known about the complexity of p -HALT (Section 3). Then we will see that the statement p -HALT $\in \text{XP}_{\text{uni}}$ is equivalent to other prominent open problems from different areas of theoretical computer science. More precisely, in Section 4 we show that it is equivalent to the existence of polynomially optimal proof systems. In Section 5 we show that p -HALT $\in \text{XP}_{\text{uni}}$ implies the existence of complete problems, first for the classical complexity class UP and then for the class of polynomial time equivalence relations under so-called equivalence reductions. We get a logic capturing the complexity class P (= PTIME) under the assumption p -HALT $\in \text{XP}_{\text{uni}}$ in Section 6. Even though we originally obtained these results using distinct arguments, in the meantime we realized that they can be obtained all by the same method: first we translate the consequences of p -HALT $\in \text{XP}_{\text{uni}}$ mentioned so far into statements on listings (that is, on effective enumerations) and then, based on p -HALT $\in \text{XP}_{\text{uni}}$, we apply a technique we call the *invariantization of listings*; this technique plays

a central role in the present exposition. One could formulate the principle underlying this technique in an abstract way, and obtain our results as instances of a general theorem. However, we do not do so. Nevertheless, in Section 7 we introduce the notion of slice-wise downward monotone parameterized problem and take a closer look on its role in the preceding results. In Section 8 we relate the assumption $p\text{-HALT} \in \text{XP}_{\text{uni}}$ to the complexity of deciding whether a hard valid first-order sentence has a proof of a given length. Finally, in Section 9 we show that $p\text{-HALT} \notin \text{XP}_{\text{uni}}$ is equivalent to the existence of hard sequences for algorithms deciding TAUT.

2 Some Preliminaries

In this section we fix some notations and recall some basic definitions.

We let $\mathbb{N}[X]$ be the set of polynomials with natural numbers as coefficients. We denote the alphabet $\{0, 1\}$ by Σ and the length of a string $x \in \Sigma^*$ by $|x|$. Let 1^n be the string consisting of n many 1s and let λ denote the empty string. We identify problems with subsets Q of Σ^* . We already remarked that the restriction to the alphabet Σ does not affect the complexity of the problem $p\text{-HALT}$. Sometimes statements containing a formulation like “there is a $d \in \mathbb{N}$ such that for all $x \in \Sigma^*$: $\dots \leq |x|^d$ ” can be wrong for $x \in \Sigma^*$ with $|x| \leq 1$. We trust the reader’s common sense to interpret such statements reasonably.

A problem $Q \subseteq \Sigma^*$ has padding if there is a function $\text{pad} : \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$ computable in logarithmic space having the following properties:

- (i) For any $x, y \in \Sigma^*$, $|\text{pad}(x, y)| > |x| + |y|$ and $(\text{pad}(x, y) \in Q \iff x \in Q)$.
- (ii) There is a logspace algorithm which, given $\text{pad}(x, y)$ recovers y .

By $\langle \dots \rangle$ we denote some standard logspace and linear time computable tupling function with logspace and linear time computable inverses.

If \mathbb{A} is a (deterministic or nondeterministic) algorithm and \mathbb{A} accepts $x \in \Sigma^*$, then we denote by $t_{\mathbb{A}}(x)$ the number of steps of a shortest accepting run of \mathbb{A} on x ; if \mathbb{A} does not accept x , then $t_{\mathbb{A}}(x) := \infty$. By convention, $\infty > n$ for $n \in \mathbb{N}$. So we can state $p\text{-HALT}$ in a more succinct way:

$p\text{-HALT}$
Instance: A nondeterministic Turing machine \mathbb{M} and 1^n
with $n \in \mathbb{N}$.
Parameter: $|\mathbb{M}|$.
Problem: Is $t_{\mathbb{M}}(\lambda) > n$?

By default, algorithms are deterministic. If an algorithm \mathbb{A} on input x eventually halts and outputs a value, we denote it by $\mathbb{A}(x)$. We use deterministic and nondeterministic Turing machines with Σ as alphabet as our basic computational model for algorithms (and we often use the notions “algorithm” and “Turing machine” synonymously). If necessary, we will not distinguish between a Turing machine and its code, a string in Σ^* . If \mathbb{M} is a deterministic or nondeterministic

Turing machine, then $L(M)$ is the language accepted by M . We use Turing machines as acceptors and transducers. Even though we use formulations like “let M_1, M_2, \dots be an enumeration of *all* polynomial time Turing machines,” from the context it will be clear that we only refer to acceptors (or that we only refer to transducers). We assume that a run of a nondeterministic Turing machine is determined by the sequence of its states.

A polynomial time deterministic or nondeterministic Turing machine M is *clocked* if (the code of) M contains a natural number $\text{time}(M)$ such that $n^{\text{time}(M)}$ is a bound for the running time of M on inputs of length n . Of course, the function $M \mapsto \text{time}(M)$ should be computable in logspace.

2.1 Parameterized Complexity

Formally, we view *parameterized problems* as pairs (Q, κ) consisting of a classical problem $Q \subseteq \Sigma^*$ and a *parameterization* $\kappa : \Sigma^* \rightarrow \mathbb{N}$, which is required to be polynomial time computable. However, we will present parameterized problems in the form we did it for p -HALT and further parameterized problems in the Introduction.

We mainly consider the classes FPT_{uni} and XP_{uni} of *uniform parameterized complexity* and sometimes the classes FPT and XP of *strongly uniform parameterized complexity*. A parameterized problem (Q, κ) is in the class FPT_{uni} (or is *uniformly fixed-parameter tractable*) if $x \in Q$ is solvable by an algorithm running in time $\leq f(\kappa(x)) \cdot |x|^{O(1)}$ for some $f : \mathbb{N} \rightarrow \mathbb{N}$. The problem (Q, κ) is in the class XP_{uni} if $x \in Q$ is solvable by an algorithm running in time $\leq |x|^{f(\kappa(x))}$ for some $f : \mathbb{N} \rightarrow \mathbb{N}$.

If in the definition of FPT_{uni} and XP_{uni} we require the function f to be computable, then we get the corresponding classes FPT and XP .

The following inclusions hold between the four complexity classes of parameterized problems just introduced:

$$\begin{array}{ccc}
 & \text{FPT}_{\text{uni}} & \\
 & \subset & \\
 \text{FPT} & & \subset \text{XP}_{\text{uni}} \\
 & \subset & \\
 & \text{XP} &
 \end{array} \tag{1}$$

While the corresponding \subseteq -inclusions are trivial, the strict inclusions $\text{FPT} \subset \text{FPT}_{\text{uni}}$ and $\text{XP} \subset \text{XP}_{\text{uni}}$ are due to Downey and Fellows [15] and to Downey [17]. The strict inclusions $\text{FPT} \subset \text{XP}$ and $\text{FPT}_{\text{uni}} \subset \text{XP}_{\text{uni}}$ are easily obtained by showing that $p\text{-EXP-DTM-HALT} \in \text{XP} \setminus \text{FPT}_{\text{uni}}$ (cf. [20, Corollary 2.26]).

3 The Complexity of p -Halt

In this section we report what we know on the complexity of p -HALT. We start with a simple observation. The problem HALT is in coNE even if the natural number n is given in binary. From that one easily obtains:

Theorem 1 ([9]). *If $E = NE$ (and hence if $P = NP$), then $p\text{-HALT} \in \text{FPT}$.*

(Assuming $E \neq NE$), by (1) the most ambitious task is to show that $p\text{-HALT} \notin \text{XP}_{\text{uni}}$ and $p\text{-HALT} \notin \text{FPT}$ should be the easiest one. We know:

Theorem 2 ([8]). *If $\text{NP}[\text{TC}] \neq \text{P}[\text{TC}]$, then $p\text{-HALT} \notin \text{FPT}$.*

Here $\text{NP}[\text{TC}] \neq \text{P}[\text{TC}]$ means that $\text{DTIME}(h^{O(1)}) \neq \text{NTIME}(h^{O(1)})$ for all time constructible and increasing functions $h : \mathbb{N} \rightarrow \mathbb{N}$. If $\text{NP}[\text{TC}] \neq \text{P}[\text{TC}]$, then $P \neq NP$, even $E \neq NE$, as seen by taking as h the identity function and the function 2^n , respectively. On the other hand, it is not hard to see (cf. [8]) that the assumption “NP contains a P-bi-immune problem” and hence the so-called Measure Hypothesis imply $\text{NP}[\text{TC}] \neq \text{P}[\text{TC}]$.

The following idea underlies a proof of Theorem 2. Assume that $p\text{-HALT} \in \text{FPT}$. Then, we have a *deterministic* algorithm deciding $p\text{-HALT}$, the parameterized halting problem for *nondeterministic* Turing machines. This yields a way (different from brute force) to translate nondeterministic algorithms into deterministic ones; a careful analysis of this translation shows that $\text{NTIME}(h^{O(1)}) \subseteq \text{DTIME}(h^{O(1)})$ for a suitable time constructible and increasing function h . For a detailed proof we refer the reader to [8].

One can refine the previous argument to get $p\text{-HALT} \notin \text{XP}$; however one needs a complexity-theoretic assumption (apparently) stronger than $\text{NP}[\text{TC}] \neq \text{P}[\text{TC}]$, namely the assumption $\text{NP}[\text{TC}] \not\subseteq \text{P}[\text{TC}^{\log \text{TC}}]$; it claims that $\text{NTIME}(h^{O(1)}) \not\subseteq \text{DTIME}(h^{O(\log h)})$ for every time constructible and increasing function $h : \mathbb{N} \rightarrow \mathbb{N}$. That is:

Theorem 3. *If $\text{NP}[\text{TC}] \not\subseteq \text{P}[\text{TC}^{\log \text{TC}}]$, then $p\text{-HALT} \notin \text{XP}$.*

The assumption “NP contains an E-bi-immune problem” implies the statement $\text{NP}[\text{TC}] \not\subseteq \text{P}[\text{TC}^{\log \text{TC}}]$.

As mentioned, we do not know whether $p\text{-HALT} \in \text{XP}_{\text{uni}}$ or whether even $p\text{-HALT} \in \text{FPT}_{\text{uni}}$. However, from the point of view of nonuniform parameterized complexity the problem $p\text{-HALT}$ is fixed-parameter tractable. Recall that a parameterized problem (Q, κ) is in the class FPT_{nu} (or is *nonuniformly fixed-parameter tractable*) if there is a constant $c \in \mathbb{N}$, an arbitrary function $f : \mathbb{N} \rightarrow \mathbb{N}$, and for every $k \in \mathbb{N}$ an algorithm solving the (classical) problem

$$(Q, \kappa)_k := \{x \in Q \mid \kappa(x) = k\}$$

in time $f(k) \cdot |x|^c$. The problem $(Q, \kappa)_k$ is called the k th *slice* of (Q, κ) .

Proposition 4. *The problem $p\text{-HALT}$ is in the class FPT_{nu} .*

Proof. Fix $k \in \mathbb{N}$; then there are only finitely many nondeterministic Turing machines \mathbb{M} with $|\mathbb{M}| = k$, say, $\mathbb{M}_1, \dots, \mathbb{M}_s$. Hence the algorithm \mathbb{A}_k that on any instance $\langle \mathbb{M}, 1^n \rangle$ of $p\text{-HALT}$ with $|\mathbb{M}| = k$ determines the i with $\mathbb{M} = \mathbb{M}_i$, and then accepts if and only if $t_{\mathbb{M}_i}(\lambda) > n$, decides the k th slice of $p\text{-HALT}$. It has running time $O(|\mathbb{M}| + n)$; thus it witnesses that $p\text{-HALT} \in \text{FPT}_{\text{nu}}$. \square

The following lemma shows that $p\text{-HALT} \in \text{XP}_{\text{uni}}$ if there is an algorithm that accepts $p\text{-HALT}$ and that runs in the time required by XP_{uni} for instances $\langle \mathbb{M}, 1^n \rangle$, where \mathbb{M} is a nondeterministic Turing machine which does not halt on the empty input tape.

Lemma 5. *If there is an algorithm \mathbb{A} accepting $p\text{-HALT}$ such that for all instances $\langle \mathbb{M}, 1^n \rangle$ with $t_{\mathbb{M}}(\lambda) = \infty$ we have $t_{\mathbb{A}}(\langle \mathbb{M}, 1^n \rangle) = n^{f(|M|)}$ for some function f , then $p\text{-HALT} \in \text{XP}_{\text{uni}}$.*

Proof. Let \mathbb{A} be as in the statement of the lemma. Let \mathbb{B} be an algorithm that on input \mathbb{M} computes $t_{\mathbb{M}}(\lambda)$ by systematically checking for $r = 0, 1, \dots$ whether there is a run of length r accepting λ . Note that \mathbb{B} does not stop on inputs \mathbb{M} with $t_{\mathbb{M}}(\lambda) = \infty$.

Now we consider the algorithm \mathbb{A}^* that on input $\langle \mathbb{M}, 1^n \rangle$ in parallel simulates \mathbb{A} on input $\langle \mathbb{M}, 1^n \rangle$ and \mathbb{B} on input \mathbb{M} . If \mathbb{A} accepts, then \mathbb{A}^* accepts. If \mathbb{B} outputs $t_{\mathbb{M}}(\lambda)$, then \mathbb{A}^* checks whether $t_{\mathbb{M}}(\lambda) > n$ and answers accordingly.

Clearly, \mathbb{A}^* decides $p\text{-HALT}$. We still have to show that it runs in time polynomial in n for fixed nondeterministic Turing machine \mathbb{M} . By our assumption on \mathbb{A} , this is clear if $t_{\mathbb{M}}(\lambda) = \infty$; if $t_{\mathbb{M}}(\lambda) < \infty$, then eventually \mathbb{B} will halt on input \mathbb{M} and output $t_{\mathbb{M}}(\lambda)$. As the check $t_{\mathbb{M}}(\lambda) > n$ can be done in linear time, in this case the running time of \mathbb{A}^* can be bounded by $O(n)$ (where the constant hidden in the Oh-notation depends on \mathbb{M}). □

Using the previous argument we show that the answer to the question “ $p\text{-HALT} \in \text{XP}_{\text{uni}}$?” would be the same if we only would require for an instance $\langle \mathbb{M}, 1^n \rangle$ of $p\text{-HALT}$ that we get the correct answer if $t_{\mathbb{M}}(\lambda)$ is not near to n . Let us give a precise version of what we mean. Let $\rho : \mathbb{N} \rightarrow \mathbb{N}$ be a nondecreasing and polynomial time computable function when inputs and outputs are given in unary notation. We say that the approximation problem $p\text{-APP-HALT}$ is in XP_{uni} if there is a function $f : \mathbb{N} \rightarrow \mathbb{N}$ and an algorithm \mathbb{A} that on every tuple $\langle \mathbb{M}, n \rangle$, where \mathbb{M} is a nondeterministic Turing machine and $n \in \mathbb{N}$, runs in time $n^{f(|M|)}$ and has the properties:

- (i) if $t_{\mathbb{M}}(\lambda) = \infty$, then \mathbb{A} accepts;
- (ii) if $t_{\mathbb{M}}(\lambda) < \infty$ and $n \leq \frac{t_{\mathbb{M}}(\lambda)}{\rho(t_{\mathbb{M}}(\lambda))}$, then \mathbb{A} accepts;
- (iii) if $t_{\mathbb{M}}(\lambda) < \infty$ and $t_{\mathbb{M}}(\lambda) \cdot \rho(t_{\mathbb{M}}(\lambda)) \leq n$, then \mathbb{A} rejects.

Thus, if $t_{\mathbb{M}}(\lambda) < \infty$, then the answer of \mathbb{A} can be arbitrary for n with

$$\frac{t_{\mathbb{M}}(\lambda)}{\rho(t_{\mathbb{M}}(\lambda))} < n < t_{\mathbb{M}}(\lambda) \cdot \rho(t_{\mathbb{M}}(\lambda)).$$

Then:

Proposition 6. *$p\text{-HALT} \in \text{XP}_{\text{uni}}$ if and only if $p\text{-APP-HALT} \in \text{XP}_{\text{uni}}$.*

Proof. Clearly, every algorithm witnessing that $p\text{-HALT} \in \text{XP}_{\text{uni}}$ shows that $p\text{-APP-HALT} \in \text{XP}_{\text{uni}}$. Conversely, let \mathbb{A} witness that $p\text{-APP-HALT} \in \text{XP}_{\text{uni}}$

and let $\langle \mathbb{M}, 1^n \rangle$ be an instance of p -HALT. We simulate \mathbb{A} on $\langle \mathbb{M}, 1^{n \cdot \rho(n)} \rangle$. If \mathbb{A} accepts, then, by (iii),

$$t_{\mathbb{M}}(\lambda) > n.$$

and hence, $\langle \mathbb{M}, 1^n \rangle \in p$ -HALT. Otherwise, we know that $t_{\mathbb{M}}(\lambda) < \infty$, and we compute $t_{\mathbb{M}}(\lambda)$ by brute force and check whether $t_{\mathbb{M}}(\lambda) > n$ or not. \square

4 Polynomially Optimal Propositional Proof Systems and p -Halt

By TAUT we denote the set of formulas of propositional logic that are tautologies. A *propositional proof system* in the sense of [14] is a polynomial time computable surjective function $p : \Sigma^* \rightarrow \text{TAUT}$. If $p(w) = \alpha$, we say that w is a p -proof of α .

Let p and p' be propositional proof systems. A *simulation from p' to p* is a polynomial time computable function $f : \Sigma^* \rightarrow \Sigma^*$ such that $p(f(w')) = p'(w')$ for all $w' \in \Sigma^*$. A propositional proof system p is *polynomially optimal* if for every propositional proof system p' there is a simulation from p' to p .

The quest for a polynomially optimal propositional proof system is an important open problem of proof theory. In [27] Krajíček and Pudlák conjectured that there is no polynomially optimal propositional proof system. It turns out that this conjecture is equivalent to p -HALT $\notin \text{XP}_{\text{uni}}$:

Theorem 7 ([10]). *There is a polynomially optimal propositional proof system if and only if p -HALT $\in \text{XP}_{\text{uni}}$.*

In the proof of the implication from right to left, we will use the following simple result. By definition a *listing* is an effective enumeration. We denote by $\text{PF}(\Sigma^*)$ and $\text{PF}(\text{TAUT})$ the set of all polynomial time computable functions from Σ^* to Σ^* and the set of all polynomial time computable functions from Σ^* to TAUT, respectively.

Proposition 8. *The following are equivalent:*

- (a) *There is a polynomially optimal propositional proof system.*
- (b) *There is a listing of $\text{PF}(\text{TAUT})$ by means of polynomial time Turing machines. By this we mean that there is a listing $\mathbb{M}_1, \mathbb{M}_2, \dots$ of polynomial time Turing machines computing functions h_1, h_2, \dots from Σ^* to TAUT such that $\text{PF}(\text{TAUT}) = \{h_i \mid i \geq 1\}$.*

Proof. (b) \Rightarrow (a): Let $\mathbb{M}_1, \mathbb{M}_2, \dots$ and h_1, h_2, \dots be as in (b). By repeating machines \mathbb{M}_i if necessary, we may assume that the function $i \mapsto \mathbb{M}_i$ is polynomial time computable. We fix a tautology α_0 and show that then the function $p : \Sigma^* \rightarrow \text{TAUT}$ is a polynomially optimal propositional proof system where

$$p(x) := \begin{cases} h_i(w), & \text{if } x = \langle i, w, c \rangle \text{ and } c \text{ is the computation of } \mathbb{M}_i \text{ on input } w \\ \alpha_0, & \text{otherwise.} \end{cases}$$

Then p is a propositional proof system: Our assumption on the function $i \mapsto \mathbb{M}_i$ and the presence of the computation c in the first case of the definition of p guarantee its polynomial time computability. Moreover, every $\alpha \in \text{TAUT}$ is in the range of p , as one of the h_i s will be the constant function with value α . Furthermore, p is polynomially optimal: If p' is a further propositional proof system, then there is an $i \geq 1$ such that $p' = h_i$. Therefore, $w \mapsto \langle i, w, c \rangle$, where c is the computation of \mathbb{M}_i on input w , is a simulation from p' to p .

(a) \Rightarrow (b): Let p be a polynomially optimal propositional proof system computed by the polynomial time Turing machine \mathbb{M} and let $\mathbb{M}_1, \mathbb{M}_2, \dots$ be a listing of $\text{PF}(\Sigma^*)$ by means of polynomial time Turing machines. If h_i denotes the function computed by \mathbb{M}_i , it is easy to verify that $\text{PF}(\text{TAUT}) = \{p \circ h_i \mid i \geq 1\}$ (here $p \circ h_i$ is the function $x \mapsto p(h_i(x))$). Then $\mathbb{M} \circ \mathbb{M}_1, \mathbb{M} \circ \mathbb{M}_2, \dots$ is the required listing, where $\mathbb{M} \circ \mathbb{M}_i$ denotes a natural polynomial time Turing machine computing $p \circ h_i$. \square

We first turn to a proof of the implication from right to left of Theorem 7. Thereby we use a technique, *the invariantization of listings*, that we shall use again and again in this paper; therefore, we give a detailed exposition here.

Lemma 9. *If $p\text{-HALT} \in \text{XP}_{\text{uni}}$, then there is a polynomially optimal propositional proof system.*

Proof. By the previous result it suffices to show that there exists a listing of $\text{PF}(\text{TAUT})$ by means of polynomial time Turing machines. However, it is undecidable whether a Turing machine computes a function from Σ^* to TAUT . So we start with a listing of $\text{PF}(\Sigma^*)$ by *clocked* polynomial time Turing machines, say

$$\mathbb{M}_1, \mathbb{M}_2, \dots \tag{2}$$

We denote by h_i the function computed by \mathbb{M}_i . Thus, $\text{PF}(\Sigma^*) = \{h_i \mid i \geq 1\}$. Using the hypothesis $p\text{-HALT} \in \text{XP}_{\text{uni}}$ we *invariantize* this listing in order to get a listing of $\text{PF}(\text{TAUT})$. For this purpose, for $h : \Sigma^* \rightarrow \Sigma^*$ and $n \geq 1$ we say that h is *n-tautological* if

$$h(w) \in \text{TAUT} \text{ for all } w \text{ with } |w| \leq n$$

and define $h^{\text{taut}} : \Sigma^* \rightarrow \Sigma^*$ by

$$h^{\text{taut}}(w) := \begin{cases} h(w), & \text{if } h \text{ is } |w|\text{-tautological} \\ \alpha_0, & \text{otherwise.} \end{cases}$$

Then

- (i) $h^{\text{taut}} : \Sigma^* \rightarrow \text{TAUT}$;
- (ii) $h^{\text{taut}} = h$ if $h : \Sigma^* \rightarrow \text{TAUT}$;
- (iii) if h is not n -tautological, then $h^{\text{taut}}(w) = \alpha_0$ for all $|w| \geq n$;
- (iv) if h is polynomial time computable, then so is h^{taut} .

Note that (iv) is an immediate consequence of (ii) and (iii). Hence,

$$h_1^{\text{taut}}, h_2^{\text{taut}}, \dots \tag{3}$$

is an enumeration of the elements of $\text{PF}(\text{TAUT})$. In fact, by (iv) all h_i^{taut} are polynomial time computable, by (i) their range is contained in TAUT , and by (ii) the enumeration contains all elements of $\text{PF}(\text{TAUT})$. But instead of the enumeration (3) of $\text{PF}(\text{TAUT})$ we aim at a listing of $\text{PF}(\text{TAUT})$ by means of *polynomial time Turing machines*. We show that there is an effective procedure assigning to every clocked polynomial time Turing machine \mathbb{M} computing some $h \in \text{PF}(\Sigma^*)$ a polynomial time Turing machine \mathbb{M}^{taut} computing h^{taut} . Then $\mathbb{M}_1^{\text{taut}}, \mathbb{M}_2^{\text{taut}}, \dots$ is the desired listing of $\text{PF}(\text{TAUT})$. For this purpose we need:

Claim: Assume that $p\text{-HALT} \in \text{XP}_{\text{uni}}$, then there is an algorithm \mathbb{B} that on input $\langle \mathbb{M}, 1^n \rangle$, where \mathbb{M} is a clocked polynomial time Turing machine computing a function $h : \Sigma^* \rightarrow \Sigma^*$, and $n \in \mathbb{N}$ decides whether h is an n -tautological in time $n^{g(|M|)}$ for some $g : \mathbb{N} \rightarrow \mathbb{N}$.

With this Claim it is straightforward to present the program of a polynomial time Turing machine \mathbb{M}^{taut} computing h^{taut} , where \mathbb{M} and h are as in the Claim. In fact, let \mathbb{M}^{taut} be the Turing machine that on input $w \in \Sigma^*$

using the algorithm \mathbb{B} of the Claim checks whether h is $|w|$ -tautological; if so, it computes and outputs $\mathbb{M}(w)$ by simulating \mathbb{M} ; otherwise, it outputs α_0 .

So it only remains to show the Claim.

Proof of the Claim: Let \mathbb{M} be a clocked polynomial time Turing machine computing a function $h : \Sigma^* \rightarrow \Sigma^*$. We show that we can decide whether h is n -tautological in the desired time by a reduction to the problem $p\text{-HALT}$. For this we assign to \mathbb{M} a polynomial time nondeterministic Turing machine \mathbb{M}^+ such that (as a first approximation) we have

$$h : \Sigma^* \rightarrow \text{TAUT} \iff \mathbb{M}^+ \text{ does not accept } \lambda. \tag{4}$$

To achieve this we take as \mathbb{M}^+ the machine that first guesses a string w , computes $\mathbb{M}(w)$ by simulating \mathbb{M} , and then checks if $\mathbb{M}(w)$ is a propositional formula; if not, it accepts, otherwise it guesses an assignment for $\mathbb{M}(w)$ and accepts if it does not satisfy $\mathbb{M}(w)$; otherwise it rejects. As \mathbb{M} runs in $\leq |w|^{\text{time}(\mathbb{M})}$ steps on input w , by standard means we can arrange \mathbb{M}^+ in such a way that for some polynomial $q \in \mathbb{N}[X]$ the machine \mathbb{M}^+ runs exactly $q(n)$ steps if in its first phase it guesses a string w of length n . As $q(n) < q(n + 1)$, we get (the fine-tuned version of (4))

$$\begin{aligned} \mathbb{M} \text{ is } n\text{-tautological (more precisely, } h \text{ is } n\text{-tautological)} \\ \iff \langle \mathbb{M}^+, 1^{q(n)} \rangle \in p\text{-HALT} \end{aligned}$$

(note that we need the assumption that \mathbb{M} is clocked to get \mathbb{M}^+ and the bound $q(n)$ in the desired effective way). As we assume that $p\text{-HALT} \in \text{XP}_{\text{uni}}$, we know

that whether $(\mathbb{M}^+, 1^{q(n)}) \in p\text{-HALT}$ may be checked in time $q(n)^{f(|\mathbb{M}^+|)}$ for some function $f : \mathbb{N} \rightarrow \mathbb{N}$. As $q(n)^{f(|\mathbb{M}^+|)} = n^{g(|\mathbb{M}^+|)}$ for suitable $g : \mathbb{N} \rightarrow \mathbb{N}$, we are done. \square

If $Q \subseteq \Sigma^*$, we write $\text{LIST}(Q)$ if there is a listing of all subsets in P of Q by means of polynomial time Turing machines. We use this listing property to prove the implication from left to right of Theorem 7. It is known that:

Theorem 10 ([32]). *There is a polynomially optimal propositional proof system if and only if $\text{LIST}(\text{TAUT})$.*

In a first step we will show:

Lemma 11. *If $\text{LIST}(\text{HALT})$, then $p\text{-HALT} \in \text{XP}_{\text{uni}}$.*

Proof. Let \mathbb{L} be a listing of the subsets in P of HALT by polynomial time Turing machines. As for every $(\mathbb{M}, 1^n) \in p\text{-HALT}$, the set $\{(\mathbb{M}, 1^n)\}$ is a subset in P of HALT , the following algorithm \mathbb{A} accepts $p\text{-HALT}$:

```

 $\mathbb{A}$  // a nondeterministic Turing machine  $\mathbb{M}$  and  $1^n$  with  $n \in \mathbb{N}$ 

1.  $\ell \leftarrow 1$ 
2. compute the  $\ell$ th machine listed by  $\mathbb{L}$ 
3.     simulate it on input  $(\mathbb{M}, 1^n)$ 
4.     if it accepts then accept
5.  $\ell \leftarrow \ell + 1$ 
6. goto 2.
    
```

We want to show that \mathbb{A} runs in time polynomial in n for fixed \mathbb{M} with $t_{\mathbb{M}}(\lambda) = \infty$. Then our claim follows from Lemma 5.

If \mathbb{M} does not halt on λ , then $\{(\mathbb{M}, 1^n) \mid n \in \mathbb{N}\}$ is a subset in P of HALT . Hence, there is a machine listed by \mathbb{L} , say the ℓ_0 th one, that decides this set. Then Lines 2-4 (for $\ell = \ell_0$) show that the running time of \mathbb{A} is polynomially bounded in n . \square

Proof of Theorem 7: It remains to show the implication from left to right (the other one has already been proved by Lemma 9). As the problem HALT , the problem TAUT is coNP -complete and both problems have padding; hence, they are polynomially isomorphic. Thus,

$$\text{LIST}(\text{TAUT}) \iff \text{LIST}(\text{HALT}). \tag{5}$$

If there is a polynomially optimal propositional proof system, then $\text{LIST}(\text{TAUT})$ by Theorem 10. Thus, $\text{LIST}(\text{HALT})$ by (5), hence $p\text{-HALT} \in \text{XP}_{\text{uni}}$ by the previous result. \square

Corollary 12. $\text{LIST}(\text{HALT}) \iff p\text{-HALT} \in \text{XP}_{\text{uni}}$.

Proof. Immediate by (5), Theorem 10, and Theorem 7. \square

Later we will use the following simple observation.

Lemma 13. *If LIST(HALT), then LIST(Q) for every $Q \in \text{coNP}$.*

Proof. More generally, we show:

Assume Q' has padding and LIST(Q'). If $Q \leq_{\text{pol}} Q'$, then LIST(Q).

Here, $Q \leq_{\text{pol}} Q'$ means that Q is polynomial time reducible to Q' . By the padding property we may assume that the polynomial time reduction $x \mapsto x'$ from Q to Q' is one-to-one and has a polynomial time computable inverse. Then, for every $X' \subseteq Q'$ in P, the set $X := \{x \mid x' \in X'\}$ is a subset in P of Q and every subset in P of Q is obtained in this way. Thus from a listing of the subsets in P of Q' , we get a listing of the subsets in P of Q . \square

5 Complete Problems and p -Halt

In this section for some “semantically defined” complexity classes of classical problems we will see that to show that they contain no complete problem is at least as hard as it is to show that $p\text{-HALT} \notin \text{XP}_{\text{uni}}$. We first deal with complete problems for the class of polynomial time decidable equivalence relations under so-called equivalence reductions (Section 5.1) and then with complete problems for the class UP under polynomial time reductions (Section 5.2).

5.1 Complete P-Equivalence Relations

Problems concerning the algorithmic properties of equivalence relations arise throughout mathematics and theoretical computer science. Examples are to decide whether two finite graphs are isomorphic or to decide whether two lists of numbers are equivalent in the sense that they represent the same set.

If E and E' are equivalence relations on Σ^* , a polynomial time reduction from E to E' (in the usual sense of complexity theory) is a polynomial time computable function f such that

$$(x, y) \in E \iff f(x, y) \in E'$$

for all $x, y \in \Sigma^*$. Often, in the context of equivalence relations the notion of *equivalence reducibility* is more natural than that of polynomial time reducibility. We say that E is *equivalence reducible to E'* and write $E \leq_{\text{eq}} E'$ if there is a polynomial time computable function $f : \Sigma^* \rightarrow \Sigma^*$ such that

$$(x, y) \in E \iff (f(x), f(y)) \in E'$$

for all $x, y \in \Sigma^*$, that is, writing xEy for $(x, y) \in E$ and similarly for E' ,

$$xEy \iff f(x)E'f(y).$$

For example, compare the meaning of both notions of reductions if E is the relation of isomorphism between finite groups and E' that of isomorphism between finite graphs.

In [21] Fortnow and Grochow asked whether the class $P(\text{eq})$ of all polynomial time equivalence relations contain a complete problem under equivalence reductions, that is, whether there is an equivalence relation $E_0 \in P(\text{eq})$ such that $E \leq_{\text{eq}} E_0$ for all $E \in P(\text{eq})$. We show:

Theorem 14. *If $p\text{-HALT} \in XP_{\text{uni}}$, then $P(\text{eq})$ contains a complete problem under equivalence reductions.*

To obtain this result we again want to use the technique of invariantization of listings. Hence, the first step yielding a proof of this theorem is a reformulation of its conclusion in terms of a listing. The bridge to listings is provided by the following result. The reader will find a proof in [3].

Proposition 15. *The following are equivalent:*

- (a) $P(\text{eq})$ contains a complete problem under equivalence reductions.
- (b) There is a listing of equivalence relations \leq_{eq} -cofinal in $P(\text{eq})$; more precisely, there is a listing E_1, E_2, \dots of elements of $P(\text{eq})$ by means of clocked polynomial time Turing machines such that for every $E \in P(\text{eq})$ there is an $i \geq 1$ such that $E \leq_{\text{eq}} E_i$.

In contrast to Proposition 8, here a listing in terms of *clocked* machines is required; in the following proof of Theorem 14 an additional argument is needed to get such machines; otherwise the proof runs along the lines of that of Lemma 9. We say that a Turing machine \mathbb{M} is a *Turing machine for tuples* if \mathbb{M} first checks whether a given input is a tuple (that is, has the form $\langle x, y \rangle$ with $x, y \in \Sigma^*$) and immediately rejects in the negative case.

Proof of Theorem 14: As it is undecidable whether a Turing machine for tuples accepts an equivalence relation we start with a listing

$$\mathbb{M}_1, \mathbb{M}_2, \dots \tag{6}$$

of all clocked polynomial time Turing machines for tuples. Hence, in general, the set $L(\mathbb{M}_i)$ of tuples accepted by \mathbb{M}_i , will not be an equivalence relation. We invariantize this listing. If T is a set of tuples and $n \in \mathbb{N}$, we say that T is an *n-equivalence relation* if the set of tuples of strings of length at most n , that is, if the set

$$\{ \langle x, y \rangle \in T \mid x, y \in \Sigma^{\leq n} \}$$

is an equivalence relation on $\Sigma^{\leq n}$. Furthermore, we set

$$T^{\text{eq}} := \{ \langle x, y \rangle \in T \mid T \text{ is a } \max\{|x|, |y|\}\text{-equivalence relation} \} \\ \cup \{ \langle x, x \rangle \mid x \in \Sigma^* \}.$$

Then

- (i) T^{eq} is an equivalence relation on Σ^* ;
- (ii) $T^{\text{eq}} = T$ if T is an equivalence relation;

- (iii) if T is not an n -equivalence relation, then T^{eq} has only finitely many equivalence classes with more than one element;
- (iv) if $T \in \text{P}$, then $T^{\text{eq}} \in \text{P}$.

Recall the listing (6) of all clocked polynomial time Turing machines for tuples. By (i)–(iv)

$$L(\mathbb{M}_1)^{\text{eq}}, L(\mathbb{M}_2)^{\text{eq}}, \dots$$

is an enumeration of $\text{P}(\text{eq})$. But we aim at a listing of (a \leq_{eq} -cofinal subset of) $\text{P}(\text{eq})$ by means of *clocked polynomial time Turing machines*. We show that there is an effective procedure assigning to every clocked polynomial time Turing machine \mathbb{M} for tuples a clocked polynomial time Turing machine \mathbb{M}^{eq} such that

$$L(\mathbb{M})^{\text{eq}} \leq_{\text{eq}} L(\mathbb{M}^{\text{eq}}).$$

Then $\mathbb{M}_1^{\text{eq}}, \mathbb{M}_2^{\text{eq}}, \dots$ is the desired listing. We will show:

Claim: Assume that $p\text{-HALT} \in \text{XP}_{\text{uni}}$, then there is an algorithm \mathbb{B} that on input $\langle \mathbb{M}, 1^n \rangle$, where \mathbb{M} is a clocked polynomial time Turing machine for tuples and $n \in \mathbb{N}$, decides whether $L(\mathbb{M})$ is an n -equivalence relation in time $n^{g(|M|)}$ for some $g : \mathbb{N} \rightarrow \mathbb{N}$.

Then it is straightforward to present the program of a clocked polynomial time Turing machine \mathbb{M}^{eq} (where \mathbb{M} is as in the Claim) with

$$L(\mathbb{M}^{\text{eq}}) := \{ \langle x, x \rangle \mid x \in \Sigma^* \} \cup \left\{ \langle \langle x, 1^s \rangle, \langle y, 1^s \rangle \rangle \mid \langle x, y \rangle \in L(\mathbb{M}), s \in \mathbb{N}, \right. \\ \left. \mathbb{B} \text{ accepts } \langle \mathbb{M}, 1^{|x|} \rangle \text{ in } \leq s \text{ steps and } \langle \mathbb{M}, 1^{|y|} \rangle \text{ in } \leq s \text{ steps} \right\}.$$

For $s \geq g(|M|)$, then the function $x \mapsto \langle x, 1^{|x|^s} \rangle$ is an equivalence reduction from $L(\mathbb{M})^{\text{eq}}$ to $L(\mathbb{M}^{\text{eq}})$. So it only remains to show the Claim.

Proof of the Claim: The proof parallels that of the claim in the proof of Lemma 9 in Section 4. Let \mathbb{M} be a clocked polynomial time machine for tuples. We assign to \mathbb{M} a polynomial time nondeterministic Turing machine \mathbb{M}^+ such that (as a first approximation) we have

$$L(\mathbb{M}) \text{ is an equivalence relation} \iff \mathbb{M}^+ \text{ does not accept } \lambda. \tag{7}$$

For this we take as \mathbb{M}^+ a machine that on empty input first guesses a string of the form $\langle r, x \rangle$, $\langle s, x, y \rangle$, or $\langle t, x, y, z \rangle$. Here r (“reflexivity”), s (“symmetry”), and t (“transitivity”) are, say, the strings 00, 01, 10, respectively. If the string has the form $\langle r, x \rangle$, then \mathbb{M}^+ simulates \mathbb{M} on input $\langle x, x \rangle$ and accepts if and only if \mathbb{M} rejects; similarly, for strings $\langle s, x, y \rangle$ the machine \mathbb{M}^+ simulates \mathbb{M} on input $\langle x, y \rangle$ and on input $\langle y, x \rangle$ and accepts if and only if \mathbb{M} accepts $\langle x, y \rangle$ and rejects $\langle y, x \rangle$; it should be clear how \mathbb{M}^+ behaves on strings of the form $\langle t, x, y, z \rangle$.

As \mathbb{M} runs in $\leq |w|^{\text{time}(\mathbb{M})}$ steps on input w , by standard means we can arrange \mathbb{M}^+ in such a way that for some polynomial $q \in \mathbb{N}[X]$ the machine \mathbb{M}^+

runs exactly $q(n)$ steps if in its first phase it guesses a string w of length n . As $q(n) < q(n + 1)$, we get (the fine-tuned version of (7))

$$L(\mathbb{M}) \text{ is an } n\text{-equivalence relation} \iff \langle \mathbb{M}^+, 1^{q(n)} \rangle \in p\text{-HALT}.$$

As we assume that $p\text{-HALT} \in \text{XP}_{\text{uni}}$, we know that $\langle \mathbb{M}^+, 1^{q(n)} \rangle \in p\text{-HALT}$ may be checked in time $q(n)^{f(|\mathbb{M}^+|)}$ for some function $f : \mathbb{N} \rightarrow \mathbb{N}$. As $q(n)^{f(|\mathbb{M}^+|)} = n^{g(|\mathbb{M}|)}$ for suitable $g : \mathbb{N} \rightarrow \mathbb{N}$, we are done. \square

5.2 UP-Complete Problems

Recall that a nondeterministic Turing machine \mathbb{M} is *unambiguous* if for every $x \in \Sigma^*$ there is at most one accepting run of \mathbb{M} on input x . UP is the class of problems accepted by an UP-machine, that is, by an unambiguous polynomial time nondeterministic Turing machine.

In this section we derive the following result showing that apparently it will be hard to show that UP contains no problem complete under polynomial time reductions. The result is due to Meßner and Torán [29] who have shown that UP contains a problem complete under polynomial time reductions if there is a polynomially optimal proof system. In virtue of Theorem 7 this is equivalent to:

Theorem 16. *If $p\text{-HALT} \in \text{XP}_{\text{uni}}$, then UP contains a problem complete under polynomial time reductions.*

In [29], the corresponding result is shown for the class of sparse sets in NP and further results of this type are given in [26,24,25]. We encourage the interested reader to apply our proof method to get these results. Therefore, we give again a quite detailed proof for UP, even though the proof just adapts the method used for equivalence relations to the present case.

Again, first we reformulate the conclusion of Theorem 16 in terms of listings. The reformulation is provided by:

Proposition 17 ([22,26]). *The following are equivalent:*

- (a) UP contains a problem complete under polynomial time reductions.
- (b) There is a listing of problems \leq_{pol} -cofinal in UP, that is, there is a listing $\mathbb{M}_1, \mathbb{M}_2, \dots$ of clocked UP-Turing machines such that for every $Q \in \text{UP}$ there is an $i \geq 1$ such that $Q \leq_{\text{pol}} L(\mathbb{M}_i)$.

Proof of Theorem 16: If \mathbb{M} is a nondeterministic Turing machine and $n \in \mathbb{N}$ we say that \mathbb{M} is *n-unambiguous* if for every $x \in \Sigma^{\leq n}$ there is at most one accepting run of \mathbb{M} on input x . We set

$$L(\mathbb{M})^{\text{unamb}} := \{x \in L(\mathbb{M}) \mid \mathbb{M} \text{ is } |x|\text{-unambiguous}\}.$$

Then

- (i) $L(\mathbb{M})^{\text{unamb}} = L(\mathbb{M})$ if \mathbb{M} is unambiguous;
- (ii) if \mathbb{M} is not n -unambiguous, then $L(\mathbb{M})^{\text{unamb}}$ contains only strings of length less than n ;
- (iii) if \mathbb{M} is a polynomial time nondeterministic Turing machine, then $L(\mathbb{M})^{\text{unamb}}$ is accepted by a UP-machine.

Let $\mathbb{M}_1, \mathbb{M}_2, \dots$ be a listing of all clocked polynomial time nondeterministic Turing machines. Then, by (i)–(iii),

$$L(\mathbb{M}_1)^{\text{unamb}}, L(\mathbb{M}_2)^{\text{unamb}}, \dots \tag{8}$$

is an enumeration of UP. But we aim at a listing of (a \leq_{pol} -cofinal subset of) UP by means of *clocked UP-machines*. We show that there is an effective procedure assigning to every clocked polynomial time nondeterministic Turing machine \mathbb{M} a clocked UP-machine $\mathbb{M}^{\text{unamb}}$ such that

$$L(\mathbb{M})^{\text{unamb}} \leq_{\text{pol}} L(\mathbb{M}^{\text{unamb}}).$$

Then $\mathbb{M}_1^{\text{unamb}}, \mathbb{M}_2^{\text{unamb}}, \dots$ is the desired listing. We will show:

Claim: Assume that $p\text{-HALT} \in \text{XP}_{\text{uni}}$, then there is an algorithm \mathbb{B} that on input $\langle \mathbb{M}, 1^n \rangle$, where \mathbb{M} is a clocked polynomial time nondeterministic Turing machine and $n \in \mathbb{N}$, decides whether $L(\mathbb{M})$ is n -unambiguous in time $n^{g(|M|)}$ for some $g : \mathbb{N} \rightarrow \mathbb{N}$.

With this Claim it is straightforward to present the program of a clocked UP-machine $\mathbb{M}^{\text{unamb}}$ (where \mathbb{M} is as in the Claim) with

$$L(\mathbb{M}^{\text{unamb}}) := \left\{ \langle x, 1^s \rangle \mid x \in L(\mathbb{M}) \text{ and } \mathbb{B} \text{ accepts } \langle \mathbb{M}, 1^{|x|} \rangle \text{ in } \leq s \text{ steps} \right\}.$$

Then, for $s \geq g(|M|)$, the function $x \mapsto \langle x, 1^{|x|^s} \rangle$ is a polynomial time reduction from $L(\mathbb{M})^{\text{unamb}}$ to $L(\mathbb{M}^{\text{unamb}})$. So it only remains to show the Claim.

Proof of the Claim: Let \mathbb{M} be a clocked polynomial time nondeterministic Turing machine. We assign to \mathbb{M} a polynomial time nondeterministic Turing machine \mathbb{M}^+ such that (as a first approximation) we have

$$\mathbb{M} \text{ is unambiguous} \iff \mathbb{M}^+ \text{ does not accept } \lambda. \tag{9}$$

For this we take as \mathbb{M}^+ a machine that on empty input first guesses strings y, c_1 , and c_2 , then it checks whether c_1 and c_2 are distinct runs of \mathbb{M} accepting y ; if so \mathbb{M}^+ accepts, else it rejects. ‘‘Fine-tuning’’ as in the proof of the corresponding claim in the proof of Theorem 14 yields the statement. \square

6 Logics Capturing P and p-Halt

We start by recalling the concepts from logic we need.

Structures. A *vocabulary* τ is a finite set of relation symbols. Each relation symbol has an *arity*. A *structure* \mathcal{A} of vocabulary τ , or τ -*structure* (or, simply structure), consists of a nonempty set A called the *universe*, and an interpretation $R^{\mathcal{A}} \subseteq A^r$ of each r -ary relation symbol $R \in \tau$. All structures are assumed to have a finite universe. To avoid technicalities we assume in this section that all structures have as universe, for some $n \in \mathbb{N}$, the set $[n] := \{1, 2, \dots, n\}$. Then, in a canonical way, we can identify structures with nonempty strings over Σ ; in particular, $|\mathcal{A}|$ for a structure \mathcal{A} is the length of the string \mathcal{A} . Moreover, then every structure \mathcal{A} has a natural ordering $<_{\mathcal{A}}$ on it.

In this section we deal with classes of structures. Thereby we always assume that all structures of a fixed class have the same vocabulary. But distinct vocabularies may correspond to distinct classes.

Logics and logics capturing P. For our purposes a *logic* L consists

- for every vocabulary τ of a set $L[\tau]$ of strings, the set of L -sentences of vocabulary τ , and of an algorithm that for every τ and every string ξ decides whether $\xi \in L[\tau]$ (in particular, $L[\tau]$ is decidable for every τ);
- of a *satisfaction relation* \models_L ; if $(\mathcal{A}, \varphi) \in \models_L$ (written: $\mathcal{A} \models_L \varphi$), then \mathcal{A} is a τ -structure and $\varphi \in L[\tau]$ for some vocabulary τ ; furthermore, for each $\varphi \in L[\tau]$ the class

$$\text{MOD}_L(\varphi) := \{\mathcal{A} \mid \mathcal{A} \models_L \varphi\}$$

of *models of φ* is closed under isomorphism.

From now on, if we say “let φ be an L -sentence,” we mean that, in addition to φ , a vocabulary τ with $\varphi \in L[\tau]$ is given.

Definition 18. Let L be a logic.

- (a) L is a *logic for P* if for all classes S of structures closed under isomorphism (with respect to structures with universe of the form $[n]$ for some $n \in \mathbb{N}$) we have

$$S \in \text{P} \iff S = \text{MOD}_L(\varphi) \text{ for some } L\text{-sentence } \varphi.$$

- (b) L *captures P* if (a) holds and if there is an algorithm \mathbb{A} deciding \models_L (that is, for every structure \mathcal{A} and L -sentence φ the algorithm \mathbb{A} decides whether $\mathcal{A} \models_L \varphi$) and if moreover, \mathbb{A} for every fixed φ runs in time polynomial in $|\mathcal{A}|$.

Hence, if L captures P, then for every L -sentence φ the algorithm \mathbb{A} witnesses that $\text{Mod}_L(\varphi) \in \text{P}$. However, we do not necessarily know ahead of time the bounding polynomial.

- (c) L *is an effectively captures P* if L captures P and if in addition to the algorithm \mathbb{A} as in (b) there is a computable function that assigns to every L -sentence φ a polynomial $q \in \mathbb{N}[X]$ such that \mathbb{A} decides whether $\mathcal{A} \models_L \varphi$ in $\leq q(|\mathcal{A}|)$ steps.

If there is no logic capturing P , then $P \neq NP$ (as Fagin [18] showed that there is a logic capturing NP). We prove that then even $p\text{-HALT} \notin XP_{\text{uni}}$ (cf. Theorem 1):

Theorem 19. *If $p\text{-HALT} \in XP_{\text{uni}}$, then there is a logic capturing P .*

In addition, we want to show that under the assumption $p\text{-HALT} \in XP_{\text{uni}}$ the so-called invariant least fixpoint logic captures P .

The following well-known result yields the desired reformulation in terms of a listing. We include a proof as we shall make use of the logic associated with a listing. Here we say that a Turing machine M is a *Turing machine for structures* if (the code of) M contains a vocabulary τ and M first checks whether a given input is a τ -structure and immediately rejects in the negative case. We then also say that M is a τ -machine.

Proposition 20. *The following are equivalent:*

- (a) *There is a logic (effectively) capturing P .*
- (b) *There is a listing of all classes in P of structures closed under isomorphism by means of (clocked) polynomial time Turing machines for structures.*

Proof. (b) \Rightarrow (a): Let the listing L with the enumeration

$$M_1, M_2, \dots$$

be as in (b). We may assume that $|M_1| < |M_2| < \dots$ by adding dummy lines to the programs of the machines if necessary. Then the logic $L(L)$ given by

$$L(L)[\tau] := \{M_i \mid i \geq 1 \text{ and } M_i \text{ is a } \tau\text{-machine}\}$$

and

$$\mathcal{A} \models_{L(L)} M \iff M \text{ accepts } \mathcal{A}$$

is a logic capturing P . If all the machines of the listing are clocked, then $L(L)$ effectively captures P .

Conversely, if L is a logic (effectively) capturing P and A is the algorithm deciding $\mathcal{A} \models_L \varphi$ in time $|\mathcal{A}|^{f(|\varphi|)}$ for some (computable) $f : \mathbb{N} \rightarrow \mathbb{N}$, denote by M_φ the (clocked) polynomial time Turing machine obtained by restricting A to inputs of the form $\langle \dots, \varphi \rangle$. Then for any effective enumeration $\varphi_1, \varphi_2, \dots$ of the sentences of L , the listing

$$M_{\varphi_1}, M_{\varphi_2}, \dots$$

is the desired listing of all classes in P of structures closed under isomorphism. \square

Proof of Theorem 19: Again we face the problem that it is undecidable whether a Turing machine for structures accepts a class closed under isomorphism. Therefore we start with a listing

$$M_1, M_2, \dots \tag{10}$$

of all clocked polynomial time Turing machines for structures. In general, the class $L(M_i) = \{\mathcal{A} \mid M_i \text{ accepts } \mathcal{A}\}$ of structures accepted by M_i will not be

closed under isomorphism. We apply the invariantization technique to get a listing as required by part (b) of Proposition 20.

If S is a class of structures and $n \in \mathbb{N}$, then we say that S is $n \cong$ -invariant if for all isomorphic structures \mathcal{A} and \mathcal{B} whose universes have at most n elements we have

$$\mathcal{A} \in S \iff \mathcal{B} \in S.$$

We set

$$S^{\text{inv}} := \{ \mathcal{A} \in S \mid S \text{ is } |\mathcal{A}| \cong\text{-invariant} \}. \tag{11}$$

We have

- (i) S^{inv} is closed under isomorphism;
- (ii) $S^{\text{inv}} = S$ if S is closed under isomorphism;
- (iii) if S is not $n \cong$ -invariant, then all structures in S^{inv} have less than n elements;
- (iv) if $S \in \mathbf{P}$, then $S^{\text{inv}} \in \mathbf{P}$.

Thus,

$$L(\mathbb{M}_1)^{\text{inv}}, L(\mathbb{M}_2)^{\text{inv}}, \dots$$

is an enumeration of all classes in \mathbf{P} of structures closed under isomorphism (in fact, the classes are in \mathbf{P} by (iv), they are closed under isomorphism by (i), and all such classes occur by (ii)).

We show that there is an effective procedure assigning to every clocked polynomial time Turing machine \mathbb{M} for structures a (clocked) polynomial time Turing machine \mathbb{M}^{inv} for structures such that

$$L(\mathbb{M}^{\text{inv}}) = L(\mathbb{M})^{\text{inv}}. \tag{12}$$

Then $\mathbb{M}_1^{\text{inv}}, \mathbb{M}_2^{\text{inv}}, \dots$ is the desired listing.

Claim: Assume that $p\text{-HALT} \in \text{XP}_{\text{uni}}$ ($p\text{-HALT} \in \text{XP}$), then there is an algorithm \mathbb{B} that on input $\langle \mathbb{M}, 1^n \rangle$, where \mathbb{M} is a clocked polynomial time Turing machine for structures and $n \in \mathbb{N}$, decides whether $L(\mathbb{M})$ is $n \cong$ -invariant in time $n^{g(|M|)}$ for some (computable) $g : \mathbb{N} \rightarrow \mathbb{N}$.

With this claim and the definition (11) of S^{inv} it is straightforward to present the program of a (clocked) polynomial time Turing machine \mathbb{M}^{inv} for structures satisfying (12).

Proof of the Claim: As in preceding proofs we reduce our problem to $p\text{-HALT}$. Let \mathbb{M} be a clocked polynomial time Turing machine for structures. We assign to \mathbb{M} a nondeterministic Turing machine \mathbb{M}^+ such that (as a first approximation) we have

$$L(\mathbb{M}) \text{ is closed under isomorphism } \iff \mathbb{M}^+ \text{ does not accept } \lambda. \tag{13}$$

For this we take as \mathbb{M}^+ a machine that on empty input first guesses strings \mathcal{A} , \mathcal{B} , and f , and accepts if \mathcal{A} and \mathcal{B} are structures, f is an isomorphism between \mathcal{A} and \mathcal{B} , and \mathbb{M} accepts \mathcal{A} but rejects \mathcal{B} .

As \mathbb{M} runs in $\leq |w|^{\text{time}(\mathbb{M})}$ steps on input w , by standard means we can arrange \mathbb{M}^+ in such a way that for some polynomial $q \in \mathbb{N}[X]$ the machine \mathbb{M}^+ runs exactly $q(n)$ steps if in its first phase it guesses a structure \mathcal{A} with n elements. As $q(n) < q(n + 1)$, we get (the fine-tuned version of (13))

$$L(\mathbb{M}) \text{ is } n \cong\text{-invariant} \iff \langle \mathbb{M}^+, 1^{q(n)} \rangle \in p\text{-HALT}.$$

As we assume that $p\text{-HALT} \in \text{XP}_{\text{uni}}$ ($p\text{-HALT} \in \text{XP}$), we know that $(\mathbb{M}^+, 1^{q(n)}) \in p\text{-HALT}$ may be checked in time $q(n)^{f(|\mathbb{M}^+|)}$ for some (computable) function $f : \mathbb{N} \rightarrow \mathbb{N}$. As $q(n)^{f(|\mathbb{M}^+|)} = n^{g(|\mathbb{M}|)}$ for a suitable (computable) $g : \mathbb{N} \rightarrow \mathbb{N}$, we are done. □

In the previous proof we have shown the implication “(a) \Rightarrow (b)” of:

Proposition 21. *The following are equivalent:*

- (a) $p\text{-HALT} \in \text{XP}_{\text{uni}}$ ($p\text{-HALT} \in \text{XP}$);
- (b) *There is an effective procedure assigning to every clocked polynomial time Turing machine \mathbb{M} for structures a (clocked) polynomial time Turing machine \mathbb{M}^{inv} for structures such that*

$$L(\mathbb{M}^{\text{inv}}) = L(\mathbb{M})^{\text{inv}}.$$

Proof. In the following we leave the proofs of the effective versions of our claims to the reader. It remains to prove the implication (b) \Rightarrow (a). For $k \geq 1$ let τ_k be the vocabulary $\{P_1, \dots, P_k\}$ with unary relation symbols P_1, \dots, P_k .

Let \mathbb{M} be a nondeterministic Turing machine. We can assume that $[k]$ for some $k \geq 1$ is the set of states of \mathbb{M} . By our convention on nondeterministic machines every two distinct successor configurations of a given configuration of \mathbb{M} have distinct states. Furthermore, here we assume that the starting state is not an accepting state. We let $S(\mathbb{M})$ be the class of τ_k -structures \mathcal{B} satisfying (i) and (ii).

- (i) The P_i 's with $i \in [k]$ form a partition of the universe B of \mathcal{B} and there is an $n \in \mathbb{N}$ such that $|P_i| = n$ for all $i \in [k]$.
- (ii) If i_1, \dots, i_n (where n is according to (i)) are such that the j th element of B in the natural ordering $<_B$ of B is in P_{i_j} , then we require that neither i_1, \dots, i_n nor any initial segment of it are the states of a run accepting the empty input.

Clearly, $S(\mathbb{M}) \in \text{P}$ and from \mathbb{M} we get a clocked polynomial time machine \mathbb{M}_0 for τ_k -structures deciding $S(\mathbb{M})$, that is,

$$L(\mathbb{M}_0) = S(\mathbb{M}).$$

Furthermore, let $\mathcal{A}(\mathbb{M}, n)$ be the τ_k -structure with universe $[n \cdot k]$ and with $P_i := \{n \cdot (i - 1) + 1, \dots, n \cdot (i - 1) + n\}$ for all $i \in [k]$. Then $\mathcal{A}(\mathbb{M}, n) \in S(\mathbb{M})$ as the starting state is not an accepting one. Furthermore,

$$\mathcal{A}(\mathbb{M}, n) \in S(\mathbb{M})^{\text{inv}} (= L(\mathbb{M}_0)^{\text{inv}}) \iff \langle \mathbb{M}, 1^n \rangle \in p\text{-HALT},$$

as we obtain all possible sequences of states of length n by considering the isomorphic copies of $\mathcal{A}(\mathbb{M}, n)$. Thus, by our assumption (b), we have for the machine $\mathbb{M}_0^{\text{inv}}$,

$$\mathbb{M}_0^{\text{inv}} \text{ accepts } \mathcal{A}(\mathbb{M}, n) \iff \langle \mathbb{M}, 1^n \rangle \in p\text{-HALT}.$$

Therefore, the following algorithm **A** shows that $p\text{-HALT} \in \text{XP}_{\text{uni}}$: On input \mathbb{M} , a nondeterministic Turing machine, and 1^n with $n \in \mathbb{N}$, it first computes the structure $\mathcal{A}(\mathbb{M}, n)$ and the clocked polynomial time machine \mathbb{M}_0 ; then applying the effective procedure of (b), it gets the machine $\mathbb{M}_0^{\text{inv}}$; finally, it checks whether $\mathbb{M}_0^{\text{inv}}$ accepts $\mathcal{A}(\mathbb{M}, n)$. \square

6.1 The Invariant Least Fixpoint Logic

Recall that in the proof of Theorem 19 we started with a listing $\mathbb{M}_1, \mathbb{M}_2, \dots$ of all clocked polynomial time Turing machine for structures and obtained under the hypothesis $p\text{-HALT} \in \text{XP}_{\text{uni}}$ ($p\text{-HALT} \in \text{XP}$) a listing $\mathbb{M}_1^{\text{inv}}, \mathbb{M}_2^{\text{inv}}, \dots$ of all classes in P of structures closed under isomorphism by means of (clocked) polynomial time Turing machines for structures. We denote this listing $\mathbb{M}_1^{\text{inv}}, \mathbb{M}_2^{\text{inv}}, \dots$ by \mathbb{L} . Then the logic $L(\mathbb{L})$, the logic assigned to \mathbb{L} in the proof of the direction “(b) \Rightarrow (a)” of Proposition 20, (effectively) captures polynomial time. In this section we present a more “logic-friendly” version of this logic.

For every vocabulary τ we let $\tau_{<} := \tau \cup \{<\}$, where $<$ is a binary relation symbol not in τ chosen in some canonical way. A logic L captures P on ordered structures if (a) and (b) of Definition 18 hold for ordered structures and classes of ordered structures. In Definition 18 (b), for fixed $\varphi \in L[\tau_{<}]$ the algorithm **A** must witness that the class of ordered models of φ is in P . It should be clear what we mean by a logic effectively capturing P on ordered structures.

Least fixpoint logic LFP is an extension of first-order logic obtained by adding an operator which allows to speak about the least fixpoint of monotone operations definable in the logic. We only need the following property of LFP.

Theorem 22. [23,34] LFP effectively captures P on ordered structures.

If S is a class of $\tau_{<}$ -structures and $n \in \mathbb{N}$, then S is n $<$ -invariant if for all τ -structures \mathcal{A} with $|A| \leq n$ and every orderings $<_1$ and $<_2$ of A we have

$$(\mathcal{A}, <_1) \in S \iff (\mathcal{A}, <_2) \in S.$$

We define the invariant least fixpoint logic LFP_{inv} by: For every vocabulary τ we set

$$\text{LFP}_{\text{inv}}[\tau] := \text{LFP}[\tau_{<}],$$

and we define the satisfaction relation by

$$\mathcal{A} \models_{\text{LFP}_{\text{inv}}} \varphi \iff (\text{MOD}_{\text{LFP}}(\varphi) \text{ is } |A| \text{ } <\text{-invariant and } (\mathcal{A}, <_A) \models_{\text{LFP}} \varphi); \tag{14}$$

recall that $<_A$ denotes the natural ordering on A .

If $\text{MOD}_{\text{LFP}}(\varphi)$ is not n $<$ -invariant, then $\text{MOD}_{\text{LFP}_{\text{inv}}}(\varphi)$ only contains structures with universe of cardinality less than n and hence is in P . Together with the fact that LFP captures P on ordered structures, this shows that LFP_{inv} is a logic for P .

In the proof of Theorem 19 we started with a listing \mathbb{L}_0 of all clocked polynomial time Turing machines for structures. As LFP captures P on ordered structures, in a certain sense the listing \mathbb{L}_0 corresponds to the sentences of LFP in the enlarged vocabularies $\tau_{<}$. We invariantized the listing \mathbb{L}_0 by using the concept of n \cong -invariance. As orderings correspond to permutations and hence to isomorphisms, in LFP_{inv} this invariantization is taken care by the definition of its semantics in (14). Hence the following result is not surprising:

Theorem 23 ([31]). (a) $p\text{-HALT} \in \text{XP}_{\text{uni}}$ if and only if LFP_{inv} captures P .
 (b) $p\text{-HALT} \in \text{XP}$ if and only if LFP_{inv} effectively captures P .

Proof. For (a) it suffices to show that LFP_{inv} captures P if and only if there is an effective procedure as stated in Proposition 21 (b).

Assume first that such a procedure exists. As LFP effectively captures P on ordered structures, for $\varphi \in \text{LFP}_{\text{inv}}[\tau] = \text{LFP}[\tau_{<}]$, we obtain a clocked polynomial time machine \mathbb{M}_φ for τ -structures with

$$L(\mathbb{M}_\varphi) := \{ \mathcal{A} \mid (\mathcal{A}, <_{\mathcal{A}}) \models_{\text{LFP}} \varphi \}.$$

Then, one easily verifies that

$$L(\mathbb{M}_\varphi)^{\text{inv}} = \text{MOD}_{\text{LFP}_{\text{inv}}}(\varphi),$$

that is, $L(\mathbb{M}_\varphi^{\text{inv}}) = \text{MOD}_{\text{LFP}_{\text{inv}}}(\varphi)$. Hence, the algorithm that on input (\mathcal{A}, φ) , first computes \mathbb{M}_φ , then $\mathbb{M}_\varphi^{\text{inv}}$ and finally simulates $\mathbb{M}_\varphi^{\text{inv}}$ on input \mathcal{A} , decides the satisfaction relation of LFP_{inv} in the desired time.

Conversely, assume that LFP_{inv} captures P . Let \mathbb{M} be a clocked polynomial time Turing machine for τ -structures. Then

$$C := \left\{ (\mathcal{B}, <) \mid \text{for some } \mathcal{A} \text{ we have: } \left((\mathcal{B}, <) \cong (\mathcal{A}, <_{\mathcal{A}}) \text{ and } \mathbb{M} \text{ accepts } \mathcal{A} \right) \right\}$$

is a class in P of ordered τ -structures closed under isomorphism (clearly, from $(\mathcal{B}, <)$ one can determine the unique structure \mathcal{A} with $(\mathcal{B}, <) \cong (\mathcal{A}, <_{\mathcal{A}})$ in polynomial time). Hence, from \mathbb{M} we effectively get a clocked polynomial time Turing machine \mathbb{M}^* with $L(\mathbb{M}^*) = C$. Furthermore, it is well-known that from a clocked polynomial time Turing machine accepting a class in P of ordered τ -structures closed under isomorphism one effectively gets an $\text{LFP}[\tau_{<}]$ -sentence φ with $C = \text{MOD}_{\text{LFP}}(\varphi)$. Now, again it is routine to verify that $L(\mathbb{M})^{\text{inv}} = \text{MOD}_{\text{LFP}_{\text{inv}}}(\varphi)$. Thus, from the algorithm deciding the satisfaction relation and witnessing that LFP_{inv} captures P , we can extract the algorithm assigning to a clocked polynomial time Turing machine \mathbb{M} a polynomial time Turing machine \mathbb{M}^{inv} with $L(\mathbb{M}^{\text{inv}}) = L(\mathbb{M})^{\text{inv}}$. \square

7 Slicewise Downward Monotone Parameterized Problems

We already mentioned in the Introduction that we do not want to present an abstract version of the invariantization technique available if $p\text{-HALT} \in \text{XP}_{\text{uni}}$ and underlying the previous proofs. However, in this section we want to point out that hidden at the core of each of these proofs is a parameterized problem which (as we show here) is in XP_{uni} if and only if $p\text{-HALT}$ is in XP_{uni} .

The problem $p\text{-HALT}$ is slicewise downward monotone. A parameterized problem (Q, κ) is *slicewise downward monotone* if Q is decidable, all elements of Q have the form $\langle x, 1^n \rangle$ with $x \in \Sigma^*$ and $n \in \mathbb{N}$, if $\kappa(\langle x, 1^n \rangle) = |x|$, and finally if the slices are downward monotone, that is, for all $x \in \Sigma^*$ and $n, n' \in \mathbb{N}$

$$\langle x, 1^n \rangle \in Q \text{ and } n' < n \text{ imply } \langle x, 1^{n'} \rangle \in Q.$$

The following slicewise downward monotone problems

$$p\text{-TAUT}, p\text{-EQUIV}, p\text{-UNAMB}, \text{ and } p\text{-}\cong\text{-INV}$$

are hidden in our considerations on polynomially optimal proof systems, P(eq)-complete problems, UP-complete problems, and logics capturing P, respectively. Here

<p>$p\text{-TAUT}$ <i>Instance:</i> A clocked polynomial time Turing machine \mathbb{M} and 1^n with $n \in \mathbb{N}$. <i>Parameter:</i> \mathbb{M}. <i>Problem:</i> Is \mathbb{M} n-tautological?</p>
<p>$p\text{-EQUIV}$ <i>Instance:</i> A clocked polynomial time Turing machine \mathbb{M} for tuples and 1^n with $n \in \mathbb{N}$. <i>Parameter:</i> \mathbb{M}. <i>Problem:</i> Is $L(\mathbb{M})$ an n-equivalence relation?</p>
<p>$p\text{-UNAMB}$ <i>Instance:</i> A clocked polynomial time nondeterministic Turing machine \mathbb{M} and 1^n with $n \in \mathbb{N}$. <i>Parameter:</i> \mathbb{M}. <i>Problem:</i> Is \mathbb{M} n-unambiguous?</p>
<p>$p\text{-}\cong\text{-INV}$ <i>Instance:</i> A clocked polynomial time Turing machine \mathbb{M} for structures and 1^n with $n \in \mathbb{N}$. <i>Parameter:</i> \mathbb{M}. <i>Problem:</i> Is $L(\mathbb{M})$ n \cong-invariant?</p>

The *Claims* in the proofs of Lemma 9, Theorem 14, Theorem 16, and Theorem 19 show that all these problem are in XP_{uni} if $p\text{-HALT} \in XP_{\text{uni}}$. We show:

Theorem 24. *If one of the problems*

$$p\text{-TAUT}, \quad p\text{-EQUIV}, \quad p\text{-UNAMB}, \quad p\text{-}\cong\text{-INV}, \quad \text{and} \quad p\text{-HALT}$$

is in XP_{uni} , then all are.

To compare the complexity of parameterized problems we use standard notions of reductions of parameterized complexity theory that we recall first. Let (Q, κ) and (Q', κ') be parameterized problems. We write $(Q, \kappa) \leq_{\text{fpt}} (Q', \kappa')$ if there is an *fpt-reduction* from (Q, κ) to (Q', κ') , that is, a mapping $R : \Sigma^* \rightarrow \Sigma^*$ with:

- (1) For all $x \in \Sigma^*$ we have $(x \in Q \iff R(x) \in Q')$.
- (2) $R(x)$ is computable in time $f(\kappa(x)) \cdot |x|^{O(1)}$ for some computable $f : \mathbb{N} \rightarrow \mathbb{N}$.
- (3) There is a computable function $g : \mathbb{N} \rightarrow \mathbb{N}$ such that $\kappa'(R(x)) \leq g(\kappa(x))$ for all $x \in \Sigma^*$.

We write $(Q, \kappa) \leq_{\text{xp}} (Q', \kappa')$ if there is an *xp-reduction* from (Q, κ) to (Q', κ') , which is defined as $(Q, \kappa) \leq_{\text{fpt}} (Q', \kappa')$ except that instead of (2) it is only required that $R(x)$ is computable in time $|x|^{f(\kappa(x))}$ for some computable $f : \mathbb{N} \rightarrow \mathbb{N}$. These are notions of reductions of the usual (strongly uniform) parameterized complexity theory. We shall use the following simple observation.

Lemma 25. *If $(Q, \kappa) \leq_{\text{xp}} (Q', \kappa')$ and $(Q', \kappa') \in XP_{\text{uni}}$, then $(Q, \kappa) \in XP_{\text{uni}}$.*

Proof of Theorem 24. By the lemma and by the remark preceding the theorem it suffices to show that

$$p\text{-HALT} \leq_{\text{fpt}} p\text{-TAUT} \leq_{\text{xp}} p\text{-EQUIV} \leq_{\text{xp}} p\text{-UNAMB} \leq_{\text{xp}} p\text{-}\cong\text{-INV}.$$

$p\text{-HALT} \leq_{\text{fpt}} p\text{-TAUT}$: Let \mathbb{M} be a nondeterministic Turing machine. We choose $s(\mathbb{M}) \in \mathbb{N}$ such that all states of \mathbb{M} are (coded by) strings of length $s(\mathbb{M})$. We fix a tautology α_0 and define $f : \Sigma^* \rightarrow \Sigma^*$ by

$$f(w) := \begin{cases} \lambda, & \text{if } w \text{ is the sequence of states of a run of } \mathbb{M} \text{ accepting } \lambda; \\ \alpha_0, & \text{otherwise.} \end{cases}$$

Of course, there is a polynomial time procedure assigning to \mathbb{M} a clocked polynomial time Turing machine \mathbb{M}' computing f . Then

$$\langle \mathbb{M}, 1^n \rangle \in p\text{-HALT} \iff \langle \mathbb{M}', 1^{n \cdot s(\mathbb{M})} \rangle \in p\text{-TAUT},$$

so that $\langle \mathbb{M}, 1^n \rangle \mapsto \langle \mathbb{M}', 1^{n \cdot s(\mathbb{M})} \rangle$ is an fpt-reduction from $p\text{-HALT}$ to $p\text{-TAUT}$.

$p\text{-TAUT} \leq_{\text{xp}} p\text{-EQUIV}$: For a clocked polynomial time Turing machine \mathbb{M} let \mathbb{M}' be the Turing machine for tuples that accepts $\langle x, y \rangle$ if either $x = y$ or $(x = \lambda \text{ and } y = \langle w, v \rangle)$, where $\mathbb{M}(w)$ is no propositional formula or $\mathbb{M}(w)$ is a

propositional formula and v a valuation which does not satisfy $\mathbb{M}(w)$). Again we can assume that from \mathbb{M} we get a clocked polynomial time such \mathbb{M}' and that for some polynomial $q \in \mathbb{N}[X]$

$$\langle \mathbb{M}, 1^n \rangle \in p\text{-TAUT} \iff \langle \mathbb{M}', 1^{q(n)} \rangle \in p\text{-EQUIV}.$$

This yields the reduction $\langle \mathbb{M}, 1^n \rangle \mapsto \langle \mathbb{M}', 1^{q(n)} \rangle$ from $p\text{-TAUT}$ to $p\text{-EQUIV}$, which is an xp-reduction and not an fpt-reduction as the degree of the polynomial q depends on $\text{time}(\mathbb{M})$.

$p\text{-EQUIV} \leq_{\text{xp}} p\text{-UNAMB}$: Let \mathbb{M} be a clocked polynomial time machine for tuples. We consider the following clocked nondeterministic polynomial time machine \mathbb{M}' : It has an initial state which is left in the first step and cannot be visited again during any run on any input. From the initial state a direct transition to an accepting state is possible (independent of the symbol scanned by the head). Hence, $L(\mathbb{M}') = \Sigma^*$. Furthermore, all other runs on inputs which do not have the form $\langle r, x \rangle$, $\langle s, x, y \rangle$, or $\langle t, x, y, z \rangle$ will be rejecting. Here r (“reflexivity”), s (“symmetry”), and t (“transitivity”) are the strings 00, 01, 10, respectively. On input $\langle r, x \rangle$ there is a run of \mathbb{M}' which simulates \mathbb{M} on input $\langle x, x \rangle$ and accepts if and only if \mathbb{M} rejects; similarly, there is an additional accepting run of \mathbb{M}' on input $\langle s, x, y \rangle$ if and only if (\mathbb{M} accepts $\langle x, y \rangle$ and rejects $\langle y, x \rangle$); finally, there is an additional accepting run of \mathbb{M}' on input $\langle t, x, y, z \rangle$ if and only if (\mathbb{M} accepts $\langle x, y \rangle$ and $\langle y, x \rangle$ but not $\langle x, z \rangle$). In particular, we see that

$$L(\mathbb{M}) \text{ is an equivalence relation on } \Sigma^* \iff \mathbb{M}' \text{ is a UP-machine.}$$

Moreover, one can arrange matters in such a way that for some polynomial $q \in \mathbb{N}[X]$ we have

$$\langle \mathbb{M}, 1^n \rangle \in p\text{-EQUIV} \iff \langle \mathbb{M}', 1^{q(n)} \rangle \in p\text{-UNAMB}.$$

$p\text{-UNAMB} \leq_{\text{fpt}} p\text{-}\cong\text{-INV}$: We assign to a string $w \in \Sigma^*$ a structure $\mathcal{A}(w)$ of vocabulary $\tau := \{<, P_0\}$, where

- the universe of $\mathcal{A}(w)$ is $[|w|]$;
- the binary $<$ is interpreted by the natural ordering on $[|w|]$;
- the unary P_0 is interpreted by the set of positions in w carrying a 0.

For $k \geq 1$ we introduce the vocabulary $\tau_k = \{U, V, P_1, \dots, P_k, R\}$ with unary relation symbols U, V, P_1, \dots, P_k and a binary R . Let \mathbb{M} be a clocked polynomial time nondeterministic Turing machine. We assume that \mathbb{M} runs exactly $n^{\text{time}(\mathbb{M})}$ steps on inputs of length n . We set $q(n) := n^{\text{time}(\mathbb{M})}$. We let \mathbb{M}' be a clocked polynomial time Turing machine for $\tau \cup \tau_k$ -structures that accepts a structure \mathcal{B} if for some $w \in \Sigma^*$ and $n := |w|$:

- (i) (the interpretation of) U and V form a partition of B ;
- (ii) the τ -reduct on U is isomorphic to $\mathcal{A}(w)$;
- (iii) the P_i 's with $i \in [k]$ form a partition of the V -part and $|P_i| = q(n)$ for all $i \in [k]$;

- (iv) R is an ordering of its field, this field is contained in the V -part and it has exactly exactly $q(n)$ elements;
- (v) if $i_1, \dots, i_{q(n)}$ are such that the m th element of the ordering R is in P_{i_m} , then $i_1, \dots, i_{q(n)}$ is the sequence of states of a run of \mathbb{M} accepting w ;
- (vi) if $j_1, \dots, j_{q(n)}$ are such that the m th element of V in the natural ordering on B is in P_{i_m} , then either $(i_1, \dots, i_{q(n)}) = (j_1, \dots, j_{q(n)})$ or $j_1, \dots, j_{q(n)}$ is not the sequence of states of a run of \mathbb{M} accepting w .

It is easy to see that \mathbb{M} is an unambiguous machine if and only if the class of structures accepted by \mathbb{M}' is closed under isomorphism. We leave the rest of the argument to the reader. □

We close this section by showing that some results we proved for p -HALT hold for all slicewise downward monotone parameterized problems. The proof of the first result is obtained by the obvious modifications in that of Proposition 4.

Proposition 26. *Every slicewise downward monotone parameterized problem is in the class FPT_{nu} .*

If (Q, κ) is slicewise downward monotone and $x \in \Sigma^*$, we set

$$s(x) := \min\{n \mid n \in \mathbb{N} \text{ and } \langle x, 1^n \rangle \notin Q\}.$$

If $\langle x, 1^n \rangle \in Q$ for all $n \in \mathbb{N}$, then we set $s(x) := \infty$. Note that for $(Q, \kappa) = p$ -HALT and every nondeterministic Turing machine we have $s(\mathbb{M}) = t_{\mathbb{M}}(\lambda)$. Hence, the following lemma generalizes Lemma 5. As its proof runs along the same lines we omit it here.

Lemma 27. *Let (Q, κ) be slicewise downward monotone. If there is an algorithm \mathbb{A} accepting (Q, κ) such that for all instances $\langle x, 1^n \rangle$ with $s(x) = \infty$ we have $t_{\mathbb{A}}(\langle x, 1^n \rangle) = n^{f(|M|)}$ for some function f , then $(Q, \kappa) \in \text{XP}_{\text{uni}}$.*

With this lemma we show the following generalization of Lemma 11, which will be used in the next section.

Lemma 28. *Let (Q, κ) be slicewise downward monotone. If $\text{LIST}(Q)$, then $(Q, \kappa) \in \text{XP}_{\text{uni}}$.*

Proof. Let \mathbb{L} be a listing of the subsets in P of Q by polynomial time Turing machines. As for every $\langle x, 1^n \rangle \in Q$, the set $\{\langle x, 1^n \rangle\}$ is a subset in P of Q , the following algorithm \mathbb{A} accepts Q :

```

A //  $x \in \Sigma^*$  and  $1^n$  with  $n \in \mathbb{N}$ 
1.  $\ell \leftarrow 1$ 
2. compute the  $\ell$ th machine listed by  $\mathbb{L}$ 
3.     simulate it on input  $\langle x, 1^n \rangle$ 
4.     if it accepts then accept
5.  $\ell \leftarrow \ell + 1$ 
6. goto 2.
    
```

We want to show that \mathbb{A} runs in time polynomial in n for fixed x with $s(x) = \infty$. Then our claim follows from Lemma 27.

If $s(x) = \infty$, then $\{\langle x, 1^n \rangle \mid n \in \mathbb{N}\}$ is a subset in P of HALT . Hence, there is a machine listed by \mathbb{L} , say the ℓ_0 th one, that decides this set. Then Lines 2–4 (for $\ell = \ell_0$) show that the running time of \mathbb{A} is polynomially bounded in n . \square

8 The Length of First-Order Proofs and p -Halt

By the undecidability of first-order logic we know that there is no computable bound on the length of shortest proofs of valid sentences of first-order logic.¹ Mathematicians' experience seems to indicate that various valid sentences φ of first-order logic only have quite long proofs, say, proofs superpolynomial in $|\varphi|$. How hard is it to decide whether such a hard valid sentence has a proof of a length less than a given bound? Corollary 32 will show that this problem is not decidable in polynomial time if $p\text{-HALT} \notin XP_{\text{uni}}$. First we have to make precise the preceding question. By “hard valid sentences” we mean valid sentences like the Four Color Theorem or Fermat's Last Theorem, but also statements like $P \neq NP$ or the Riemann Hypothesis. Of course, we do not know whether these last two statements are valid sentences; hence the following promise problem could be viewed as the appropriate precise version of our question (note that its promise is equivalent to assuming that either φ is not valid or that φ is valid and has no short proof). Let $\iota : \mathbb{N} \rightarrow \mathbb{N}$ be a nondecreasing, unbounded, and computable function.

PROMISE-EXP-GÖDEL $_{\iota}$

Instance: A first-order sentence φ having no proof of length $< |\varphi|^{\iota(|\varphi|)}$ and 1^n with $n \geq |\varphi|^{\iota(|\varphi|)}$.

Problem: Does every proof of φ have length $> n$?

One could also consider the following (plain) problem.

EXP-GÖDEL $_{\iota}$

Instance: A first-order sentence φ and 1^n with $n \geq |\varphi|^{\iota(|\varphi|)}$.

Problem: Does every proof of φ have length $> n$?

Clearly, EXP-GÖDEL $_{\iota}$ is in coNP ; Buhrman and Hitchcock [2] have shown that sparse problems are not coNP -hard unless the polynomial hierarchy collapses. This implies (cf. [8]):

¹ Here we refer to any reasonable sound and complete proof calculus for first-order logic. However, we do not allow proof calculi, which admit all first-order instances of propositional tautologies as axioms (as then it would be difficult to recognize correct proofs if $P \neq NP$).

Lemma 29. *Assume that the polynomial hierarchy does not collapse. Then the problems PROMISE-EXP-GÖDEL_ι and EXP-GÖDEL_ι are not coNP-hard (for the problem PROMISE-EXP-GÖDEL_ι this means that the set of instances of the problem that satisfy the promise and are positive instances is not coNP-hard).*

If the two problems are not coNP-hard, how do we convince ourselves that the two problems are intractable? For this purpose we consider a further slicewise downward monotone parameterized problem, namely

p-GÖDEL
Instance: A first-order sentence φ and 1^n with $n \in \mathbb{N}$.
Parameter: $|\varphi|$.
Problem: Does every proof of φ have a length $> n$?

We establish the following relationship to the previous problems:

Proposition 30 ([8]). *Let ι be a nondecreasing, unbounded, and computable function. If PROMISE-EXP-GÖDEL_ι or EXP-GÖDEL_ι is decidable in polynomial time, then *p*-GÖDEL \in FPT.*

Proof. Assume that the algorithm \mathbb{A} decides PROMISE-EXP-GÖDEL_ι in polynomial time. Then the following algorithm \mathbb{G} shows that *p*-GÖDEL \in FPT: Given an arbitrary instance $\langle \varphi, 1^n \rangle$ of *p*-GÖDEL, by brute force \mathbb{G} checks whether a shortest proof of φ has length $s(\varphi) < |\varphi|^{\iota(|\varphi|)}$; if so, it checks whether $s(\varphi) > n$ or not and answers accordingly; otherwise, if $n < |\varphi|^{\iota(|\varphi|)}$, it accepts and if $n \geq |\varphi|^{\iota(|\varphi|)}$ (and hence the promise of PROMISE-EXP-GÖDEL_ι \in P is fulfilled), it simulates \mathbb{A} on $\langle \varphi, 1^n \rangle$ and answers accordingly.

As the “brute force check” can be done in time $\leq f(|\varphi|)$ for a suitable computable f , the algorithm \mathbb{G} witnesses that *p*-GÖDEL \in FPT. □

We show:

Theorem 31. *p -GÖDEL \in XP_{uni} if and only if p -HALT \in XP_{uni}.*

From the two previous results we get:

Corollary 32. *If p -HALT \notin XP_{uni}, then the problems PROMISE-EXP-GÖDEL_ι and EXP-GÖDEL_ι are not polynomial time decidable.*

Proof of Theorem 31. Assume first that p -HALT \in XP_{uni}. Then LIST(HALT) (by Corollary 12) and thus, by Lemma 13, LIST(GÖDEL), where GÖDEL denotes the classical problem underlying *p*-GÖDEL. Therefore, *p*-GÖDEL \in XP_{uni} by Lemma 28.

Now assume that *p*-GÖDEL \in XP_{uni}. By standard means one can show (e.g., [8, Lemma 7]) that there exists a $d \in \mathbb{N}$ and a polynomial time algorithm that assigns to every nondeterministic Turing machine \mathbb{M} a first-order sentence $\varphi_{\mathbb{M}}$ such that for $n \in \mathbb{N}$

$$\langle \varphi_{\mathbb{M}}, 1^{n^d} \rangle \in p\text{-GÖDEL} \implies \langle \mathbb{M}, 1^n \rangle \in p\text{-HALT}. \tag{15}$$

Moreover,

$$\varphi_{\mathbb{M}} \text{ has a proof } \implies \mathbb{M} \text{ accepts the empty input tape.} \tag{16}$$

Now assume that \mathbb{G} is an algorithm that witnesses $p\text{-GÖDEL} \in \text{XP}_{\text{uni}}$. Let $d \in \mathbb{N}$ be as above. We present an algorithm \mathbb{A} showing that $p\text{-HALT} \in \text{XP}_{\text{uni}}$. On an instance $\langle \mathbb{M}, 1^n \rangle$ of $p\text{-HALT}$ the algorithm \mathbb{A} first computes $\varphi_{\mathbb{M}}$ and then runs two algorithms in parallel:

- an algorithm that on input \mathbb{M} , by brute force, computes $t_{\mathbb{M}}(\lambda)$ (the least n such that \mathbb{M} on empty input tape has an accepting run of length n);
- the algorithm \mathbb{G} on input $\langle \varphi_{\mathbb{M}}, 1^{n^d} \rangle$.

If the brute force algorithm halts outputting $t_{\mathbb{M}}(\lambda)$, then \mathbb{A} checks whether $n < t_{\mathbb{M}}(\lambda)$, answers accordingly, and halts. Assume now that \mathbb{G} halts. If \mathbb{G} accepts $\langle \varphi_{\mathbb{M}}, 1^{n^d} \rangle$ (and hence $\langle \mathbb{M}, 1^n \rangle \in p\text{-HALT}$ by (15)), then \mathbb{A} accepts. If \mathbb{G} rejects $\langle \varphi_{\mathbb{M}}, 1^{n^d} \rangle$, then \mathbb{A} continues the simulation of the “brute force algorithm.”

The algorithm \mathbb{A} decides $p\text{-HALT}$: note that if \mathbb{G} rejects $\langle \varphi_{\mathbb{M}}, 1^{n^d} \rangle$, then $\langle \varphi_{\mathbb{M}}, 1^{n^d} \rangle \notin p\text{-GÖDEL}$; in particular, $\varphi_{\mathbb{M}}$ has a proof, and therefore \mathbb{M} accepts the empty input tape by (16), so that in this case the computation of the brute force algorithm eventually will output $t_{\mathbb{M}}(\lambda)$, and \mathbb{A} will answer correctly.

We still have to show that for fixed nondeterministic Turing machine \mathbb{M} the algorithm \mathbb{A} runs in time polynomial in n on inputs of the form $\langle \mathbb{M}, 1^n \rangle$. We consider two cases.

\mathbb{M} halts on empty input tape: Then an upper bound for the running time is given by the time that the brute force algorithm needs to compute $t_{\mathbb{M}}(\lambda)$ (and the time for the check whether $n < t_{\mathbb{M}}(\lambda)$); hence we have an upper bound of the form $n^{c_{\mathbb{M}}}$.

\mathbb{M} does not halt on empty input tape: Then, by (16), we have $\langle \varphi_{\mathbb{M}}, 1^{n^d} \rangle \in p\text{-GÖDEL}$; hence an upper bound is given by the running time of \mathbb{G} on input $\langle \varphi_{\mathbb{M}}, 1^{n^d} \rangle$. □

Similarly as we did for $p\text{-HALT}$ at the end of Section 3, one can show that the answer to the question “ $p\text{-GÖDEL} \in \text{XP}_{\text{uni}}?$ ” would be the same if we only would require for an instance $\langle \varphi, 1^n \rangle$ of $p\text{-GÖDEL}$ that we get the correct answer if $s(\varphi)$, the length of a shortest proof of φ , is not near to n .

9 Hard Sequences for Algorithms and $p\text{-Halt}$

Recall that an algorithm \mathbb{O} deciding a problem $Q \subseteq \Sigma^*$ is *almost optimal* if for every algorithm \mathbb{A} deciding Q there is a polynomial $p_{\mathbb{A}} \in \mathbb{N}[X]$ such that for every $x \in Q$

$$t_{\mathbb{O}}(x) \leq p_{\mathbb{A}}(t_{\mathbb{A}}(x) + |x|). \tag{17}$$

Note that nothing is required for $x \notin Q$.

In [27] it was shown that

TAUT has an almost optimal algorithm \iff
 there is a polynomially optimal propositional proof system.

Hence, by Theorem 7,

TAUT has an almost optimal algorithm $\iff p\text{-HALT} \in \text{XP}_{\text{uni}}$. (18)

Let \mathbb{A} be an algorithm deciding a problem Q . A sequence $(x_s)_{s \in \mathbb{N}}$ of strings x_s in Q is *hard for \mathbb{A}* if the function $1^s \mapsto x_s$ is computable in polynomial time and the sequence $(t_{\mathbb{A}}(x_s))_{s \in \mathbb{N}}$ is not polynomially bounded in s . Clearly, if \mathbb{A} is polynomial time, then \mathbb{A} has no hard sequences. Furthermore, an almost optimal algorithm for Q has no hard sequences either. In fact, if $(x_s)_{s \in \mathbb{N}}$ is a hard sequence for an algorithm, then one can polynomially speed up it on $\{x_s \mid s \in \mathbb{N}\}$, so it cannot be almost optimal. We show:

Theorem 33. *Every algorithm deciding TAUT has a hard sequence if and only if $p\text{-HALT} \notin \text{XP}_{\text{uni}}$.*

Proof. If $p\text{-HALT} \in \text{XP}_{\text{uni}}$, then TAUT has an almost optimal algorithm (by (18)); we have just remarked that an almost optimal algorithm has no hard sequence.

It remains to show the implication from right to left. So assume that $p\text{-HALT} \notin \text{XP}_{\text{uni}}$. Then, by Lemma 5, for every algorithm \mathbb{A} deciding $p\text{-HALT}$ there is a nondeterministic machine $\mathbb{M}(\mathbb{A})$ with $t_{\mathbb{M}(\mathbb{A})}(\lambda) = \infty$ such that \mathbb{A} restricted to instances of the form $\langle \mathbb{M}(\mathbb{A}), 1^n \rangle$ is not polynomial time.

Now let \mathbb{C} be any algorithm deciding TAUT and let \mathbb{S} be a polynomial time reduction from HALT to TAUT. Then the algorithm $\mathbb{C} \circ \mathbb{S}$ that on input x first computes $\mathbb{S}(x)$ and then simulates \mathbb{C} on input $\mathbb{S}(x)$, decides HALT. By the previous observation, $\mathbb{C} \circ \mathbb{S}$ restricted to instances of the form $\langle \mathbb{M}(\mathbb{C} \circ \mathbb{S}), 1^n \rangle$ is not polynomial time; hence, \mathbb{C} restricted to instances of the form $\mathbb{S}(\langle \mathbb{M}(\mathbb{C} \circ \mathbb{S}), 1^n \rangle)$ is not polynomial time. As $1^s \mapsto \mathbb{S}(\langle \mathbb{M}(\mathbb{C} \circ \mathbb{S}), 1^s \rangle)$ is computable in polynomial time, the sequence $(\mathbb{S}(\langle \mathbb{M}(\mathbb{C} \circ \mathbb{S}), 1^s \rangle))_{s \in \mathbb{N}}$ is a hard sequence for \mathbb{C} . \square

10 Summary, Generalizations and Extensions of the Results

Summarizing we present a theorem which contains statements from different areas of theoretical computer science we have shown to be equivalent to $p\text{-HALT} \in \text{XP}_{\text{uni}}$.

Theorem 34. *The following are equivalent:*

- (1) $p\text{-HALT} \in \text{XP}_{\text{uni}}$;
- (2) *There is a polynomially optimal propositional proof system;*
- (3) LFP_{inv} captures P ;

- (4) p -GÖDEL \in XP_{uni};
 (5) There are algorithms deciding TAUT without hard sequences.

In this expository article we only derived consequences of or statements equivalent to “ p -HALT \in XP_{uni}.” There are various extensions of these equivalences, which arise from questions like “what do p -HALT \in XP, p -HALT \in FPT, or p -HALT \in FPT_{uni} mean for these related problems?” Further complexity classes have been considered in [12].

Here we report what the effect of changing membership of p -HALT in the class XP_{uni} by a different class means for the equivalence (18). By this equivalence, p -HALT \in XP_{uni} if and only if there is an algorithm \mathbb{O} deciding TAUT such that for every further algorithm \mathbb{A} deciding TAUT there is a polynomial $p_{\mathbb{A}} \in \mathbb{N}[X]$ such that for every tautology α

$$t_{\mathbb{O}}(\alpha) \leq p_{\mathbb{A}}(t_{\mathbb{A}}(\alpha) + |\alpha|). \quad (19)$$

The statement p -HALT \in XP is equivalent to the existence of an effective procedure assigning to an algorithm \mathbb{A} deciding TAUT a polynomial $p_{\mathbb{A}}$ satisfying (19). And p -HALT \in FPT_{uni} means that for some d the polynomials $p_{\mathbb{A}}$ may be chosen of degree $\leq d$. If, in addition, they may be chosen effectively, this means that p -HALT \in FPT.

References

1. Aumann, Y., Dombb, Y.: Fixed Structure Complexity. In: Grohe, M., Niedermeier, R. (eds.) IWPEC 2008. LNCS, vol. 5018, pp. 30–42. Springer, Heidelberg (2008)
2. Buhrman, H., Hitchcock, J.M.: NP-hard sets are exponentially dense unless $\text{coNP} \subseteq \text{NP/poly}$. In: Proceedings of the 23rd Annual IEEE Conference on Computational Complexity (CCC 2008), Electronic Colloquium on Computational Complexity (ECCC 2008), Report TR08 022, pp. 1–7 (2008), <http://eccc.hpi-web.de/eccc-local/Lists/TR-2008.html>
3. Buss, S., Chen, Y., Flum, J., Friedman, S., Müller, M.: Strong isomorphism reductions in complexity theory. The Journal of Symbolic Logic 76(4), 1381–1402 (2011)
4. Cai, L., Chen, J., Downey, R., Fellows, M.: the parameterized complexity of short computation and factorization. Archive for Mathematical Logic 36, 321–337 (1997)
5. Cesati, M.: The Turing way to parameterized complexity. Journal of Computer and System Sciences 67, 654–685 (2003)
6. Cesati, M., Di Ianni, M.: Computation models for parameterized complexity. Mathematical Logical Quarterly 43, 179–202 (1997)
7. Chandra, A.K., Kozen, D., Stockmeyer, L.J.: Alternation. Journal of the ACM 28, 114–133 (1981); 77–90 (1977)
8. Chen, Y., Flum, J.: On the complexity of Gödel’s proof predicate. The Journal of Symbolic Logic 75, 239–254 (2009)
9. Chen, Y., Flum, J.: A Logic for PTIME and a Parameterized Halting Problem. In: Blass, A., Dershowitz, N., Reisig, W. (eds.) Gurevich Festschrift. LNCS, vol. 6300, pp. 251–276. Springer, Heidelberg (2010)

10. Chen, Y., Flum, J.: On p -Optimal Proof Systems and Logics for PTIME. In: Abramsky, S., Gaville, C., Kirchner, C., Meyer auf der Heide, F., Spirakis, P.G. (eds.) ICALP 2010, Part II. LNCS, vol. 6199, pp. 321–332. Springer, Heidelberg (2010)
11. Chen, Y., Flum, J.: On Slicewise Monotone Parameterized Problems and Optimal Proof Systems for TAUT. In: Dawar, A., Veith, H. (eds.) CSL 2010. LNCS, vol. 6247, pp. 200–214. Springer, Heidelberg (2010)
12. Chen, Y., Flum, J.: Listings and logics. In: Proceedings of the 26th Annual IEEE Symposium on Logic in Computer Science (LICS 2011), pp. 165–174 (2011)
13. Cook, S.: The complexity of theorem proving procedures. In: Proceedings of the Third Annual ACM Symposium on Theory of Computing, pp. 151–158 (1971)
14. Cook, S., Reckhow, R.: The relative efficiency of propositional proof systems. The Journal of Symbolic Logic 44, 36–50 (1979)
15. Downey, R., Fellows, M.: Fixed-parameter tractability and completeness III: Some structural aspects of the W -hierarchy. In: Ambos-Spies, K., et al. (eds.) Complexity Theory, pp. 166–191 (1993)
16. Downey, R., Fellows, M.: Parameterized Complexity. Springer (1999)
17. Downey, R.: Private communication
18. Fagin, R.: Generalized first-order spectra and polynomial-time recognizable sets. In: Karp, R.M. (ed.) Complexity of Computation. SIAM-AMS Proceedings, vol. 7, pp. 43–73 (1974)
19. Flum, J., Grohe, M.: Fixed-parameter tractability, definability, and model checking. SIAM Journal on Computing 31, 113–145 (2001)
20. Flum, J., Grohe, M.: Parameterized Complexity Theory. Springer (2006)
21. Fortnow, L., Grochow, J.: Complexity classes of equivalence problems revisited, arXiv:0907.4775v1, [cs.CC] (2009)
22. Hartmanis, J., Hemachandra, L.: Complexity classes without machines: On complete languages for UP. Theoretical Computer Science 58, 129–142 (1988)
23. Immerman, N.: Relational queries computable in polynomial time. Information and Control 68, 86–104 (1986)
24. Köbler, J., Messner, J.: Complete problems for promise classes by optimal proof systems for test sets. In: Proceedings of the 13th IEEE Conference on Computational Complexity (CCC 1998), pp. 132–140 (1998)
25. Köbler, J., Messner, J., Torán, J.: Optimal proof systems imply complete sets for promise classes. Information and Computation 184, 71–92 (2003)
26. Kowalczyk, W.: Some Connections Between Presentability of Complexity Classes and the Power of Formal Systems of Reasoning. In: Chytil, M.P., Koubek, V. (eds.) MFCS 1984. LNCS, vol. 176, pp. 364–369. Springer, Heidelberg (1984)
27. Krajíček, J., Pudlák, P.: Propositional proof systems, the consistency of first order theories and the complexity of computations. The Journal of Symbolic Logic 54, 1063–1088 (1989)
28. Levin, L.: Universal search problems. Problems of Information Transmission 9(3), 265–266 (1973) (in Russian); (english translation) Trakhtenbrot, B.A.: A survey of Russian approaches to perebor (brute-force search) algorithms. Annals of the History of Computing 6(4), 384–400 (1984)
29. Messner, J., Torán, J.: Optimal Proof Systems for Propositional Logic and Complete Sets. In: Meinel, C., Morvan, M. (eds.) STACS 1998. LNCS, vol. 1373, pp. 477–487. Springer, Heidelberg (1998)

30. Monroe, H.: Speedup for natural problems and noncomputability. *Theoretical Computer Science* 412(4-5), 478–481 (2011)
31. Nash, A., Rimmel, J.B., Vianu, V.: PTIME Queries Revisited. In: Eiter, T., Libkin, L. (eds.) *ICDT 2005*. LNCS, vol. 3363, pp. 274–288. Springer, Heidelberg (2005)
32. Sadowski, Z.: On an optimal propositional proof system and the structure of easy subsets. *Theoretical Computer Science* 288(1), 181–193 (2002)
33. Turing, A.: On computable numbers, with an application to the Entscheidungsproblem. *Proc. London Math. Society* 2, 230–265 (1936)
34. Vardi, M.Y.: The complexity of relational query languages. In: *Proceedings of the 14th Annual ACM Symposium on Theory of Computing (STOC 1982)*, pp. 137–146 (1982)

Computer Science Unplugged and Related Projects in Math and Computer Science Popularization

Tim Bell^{1,*}, Frances Rosamond², and Nancy Casey³

¹ Department of Computer Science and Software Engineering
University of Canterbury

Christchurch, NZ

`tim.bell@canterbury.ac.nz`

² School of Engineering and Information Technology

Charles Darwin University

Darwin, Northern Territory 0909 Australia

`Frances.Rosamond@CDU.edu.au`

³ Logwood Stone

Moscow, Idaho

`Nancy@logwoodstone.com`

Abstract. *Mathematics popularization is an important, creative kind of research, entangled with many other research programs of basic interest*

— Mike Fellows

This chapter is a history of the Computer Science Unplugged project, and related work on math and computer science popularization that Mike Fellows has been a driving force behind, including *MEGA-Mathematics* and games design. Mike’s mission has been to open up the knowns and unknowns of mathematical science to the public. We explore the genesis of *MEGA-Math* and “Unplugged” in the early 1990s, and then the sudden growth of interest in Unplugged after the year 2003, including the contributions from many different cultures and its deployment in a large variety of contexts. Woven through this history is the importance of story: that presenting math and computing topics through story-telling and drama can captivate children and adults alike, and provides a whole new level of engagement with what can be perceived as a dry topic. It is also about not paying attention to boundaries — whether teaching advanced computer science concepts to elementary school children or running a mathematics event in a park.

*Dedicated to Mike Fellows
on the occasion of his 60th birthday.*

1 Introduction

It is quite uncommon for a world class research scientist to also be heavily involved in K-12 education. It is rarer still for such a scientist to be involved in

* Corresponding author.

developing new programs and methodologies for popularizing and teaching basic principles in education.

Mike Fellows and some of his co-workers initiated just such a program beginning in the 1980's. They were deeply concerned with the trends in mathematics and computer science education. The plan was to develop materials based around modern research in computer science and mathematics and have these ideas used to make early education more exciting and engaging.

As we will describe in later sections of this paper, these ideas involved things like NP-completeness, parallel sorting networks, automata, and many other mainstays of modern computing research. The early material evolved from various projects and groups (including "Family math", "Mathmania") culminating in the "MEGA-math" project, which ultimately led to the "Computer Science Unplugged" project that has become widely used internationally.

In this paper, we will describe the projects involved and look at how the material evolved. A subtext of the paper is the difficulty of getting the material accepted, necessitating a long and hard fought campaign for eventual adoption. We also track the chance meetings and collaborations that brought about a rich and diverse range of material and events under the "CS Unplugged" umbrella.

Acceptance of Mike's efforts in mathematical outreach had a bit of a slow start, somewhat similar to the slow acceptance by theory researchers of parameterized complexity. Mike has told about publishers in the 1990's who rejected *Computer Science Unplugged* saying, "It is not a math book, and since it is 'unplugged,' we cannot publish it as a computer science book either. But, my wife loves it and so does her friend who is a teacher, so would you please send us another copy." It was quite a different scene at the 2007 ACM Special Interest Group in Computer Science Education (SIGCSE) Conference. Carnegie Mellon Computer Science Department Chair Jeannette Wing had just finished her insightful description of "computational thinking" when Lenore Blum promptly announced, "We are fortunate to have someone here who has written materials that exemplify computational thinking. Would Michael Fellows please stand up." The SIGCSE workshop featuring *Unplugged* overflowed the venue, and had to be repeated, with standing-room only.

The prehistory of *Unplugged* began with "Family Math" in the seventies, which was part of the inspiration for the *MEGA-Math* program that started to take shape in the late eighties. This happened in the United States, at a time when many parents, often educated professionals, organized "free schools," hiring teachers and volunteering at the schools. Mike attributes his early popularizing efforts to volunteering in the elementary classrooms of his children Hannah and Max at Apple Blossom Family School in Moscow, Idaho. Mike recalls hurrying from his job at the university to the primary school. He had just been teaching a topic on sorting in a graduate class, and decided to teach the same topic to the children. It was a huge success, and the beginning of many of the activities that now are mainstays of *MEGA-Math* and *Computer Science Unplugged*.

As Rod says in his chapter, Mike's homes were always full of books from all kinds of areas of human endeavor, and papers from many areas of science. Furthermore, Mike loves to talk with people from all backgrounds, about almost any subject. Rudolf Fleischer calls his thinking "top down," in the sense that he can see quickly right to the heart of an issue — and recall all the details, no matter how many years have passed. These aptitudes coupled with vast imagination and an indignation that children were being given short-shrift by the school curriculum, resulted in four Cowboy Melodramas — satires about mathematics education, that were presented at the 1998 Fringe Theatre in Victoria. Each play dramatizes the proof of a mathematical theorem, and Mike considered the plays an experiment in presenting technical information to the public. He called them "content-driven" theatre. These are discussed in more detail in the chapter on "Passion Plays," but they are closely related to the Unplugged project because they make strong use of imaginative stories to illustrate a deep point; and they directly address the issue that young students should be exposed to the exciting parts of our discipline.

As computer games became increasingly popular, Mike thought about how they could be used to convey mathematics to children, and he developed a systematic method of creating computer games harnessing the intractable computational problems in the compendium of Garey and Johnson [1].

These various experiences came together to produce Computer Science Unplugged, a collection of stories, games, puzzles, and tricks, presented as activities, shows and videos, which have led to the *Unplugged* approach to pedagogy that has become a meme in the world of computer science education.

This history of Unplugged is timely because the year 2012 marks the 20th anniversary of the publication of the book *This is MEGA-Mathematics!*, which was the seed that led to the collaboration that became known as "Computer Science Unplugged." The Unplugged project began out of an interest in providing engaging and accessible ways of introducing children to big ideas from mathematics and computer science, and has grown from a few chance collaborations into an approach to outreach and teaching with direct contributions from dozens of academics. It has been translated into about 16 languages, used in classrooms in many countries, and found its way into creative endeavors tangential to mathematics and computer science.

In the spirit of the style of teaching described in this chapter, we begin in Section 2 by diving headfirst into describing the sorting network as an example of the type of activity that was used as a tool for engaging young children with advanced ideas from math and computer science. The remainder of the chapter is organized somewhat chronologically. Section 3 reviews some of the activism that Mike engaged in to change attitudes amongst educators, one product of which was the *MEGA-Math* workbook, which is described in Section 4. Section 5 describes how this led into Computer Science Unplugged, and Section 6 explains how it suddenly gained momentum around 2003 to 2006. The general principles that emerged through this work are captured in Section 7, and evaluations of Unplugged are reviewed in Section 8. In parallel to the work on *MEGA-Math*

and Unplugged, Mike was working on computer games and the mathematical “passion plays”. Section 9 reflects on Mike’s insights into computer games; the separate chapter on passion plays describes the Cowboy Melodramas of Mathematics.

The information in this chapter has been gathered by personal recollection of many of the people involved, and a survey of the large number of writings that these projects generated. By chance, two natural disasters also provided source material for us: Mike and Frances’ papers arrived in Darwin from Newcastle, delayed by the 2010–2011 Queensland floods, just in time to use in writing this chapter, and Tim Bell’s office had to be completely emptied after the Christchurch earthquake of February 2011, bringing to the surface some early documents relating to Unplugged. Writing this chapter involved something of an archaeological dig through the resulting material, unearthing all sorts of relics from the last 20 years!

We know that the ideas have had lasting influence. It is moving to Mike, to receive on a fairly regular basis, an email from an unknown person saying something like, “Dear Dr. Fellows, I have found your article about Unplugged or *MEGA-Math* or . . . , and I just want you to know how much it means to me” In fact, Elena Prieto, who subsequently became Mike’s Ph.D. student, knocked on his University of Victoria (UVic) office door to say, “I just want to shake the hand of one of the authors of Computer Science Unplugged.” Elena had been teaching for an NGO as a mathematics lecturer at the National University of El Salvador, and relied on Unplugged when the power went out, which was not infrequent. This chapter reveals many more unexpected applications and situations where Mike’s vision has changed the way math and computing is taught!

2 The Sorting Network

The quintessential “Unplugged” activity that has been an instant hit with all ages for the last two decades is the “Sorting Network”, where a layout like the one in Figure 1(a) is drawn on the pavement in chalk (Figure 1(b)) or on an indoor surface with painter’s tape. Six students holding numbers start in the six boxes on the left, and move to the right following their respective arrow until they meet another student at a circle (node). At the circle, two students compare numbers, and the student with the smaller one follows the arrow to their left, while the student with the larger number follows the arrow to their right. Each student arrives in a new circle where they again compare numbers with the student they meet there. This structure is called a *parallel* sorting network because there are three comparisons happening at the same time.

Students and teachers alike are generally surprised when they come out the end of the network with the numbers they are holding in ascending order, and suddenly everyone is plunged into the kind of observation, questioning, critical thinking and reflection that is at the core of mathematics and computer science. Does it work backwards? Can you sort in descending order? Can you use it to

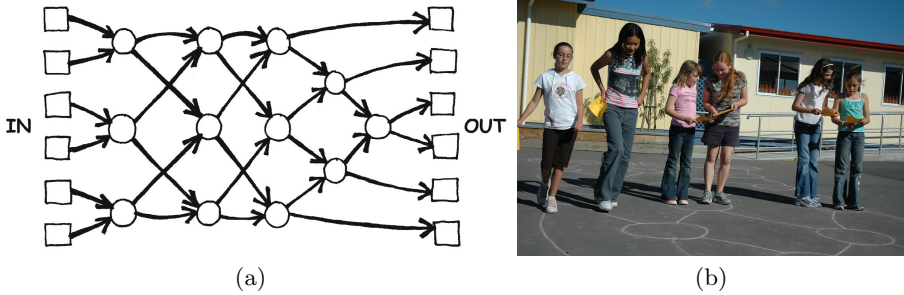


Fig. 1. (a) A 6-input parallel sorting network (b) Chalked in a school playground

sort 12 numbers? Or 7? Can you give it a set of numbers that will make it fail? Can you find a shorter network that still sorts the numbers? How many permutations would you have to test to check every possible input? Some students might immediately insist on trying it again. Others might try drawing their own networks, or begin inventing/drawing something that only they can understand. Some students simply listen and watch. While the class as a whole mimics a mathematical community whose members are grappling with different aspects of an intriguing problem, it's likely that neither the students nor teachers will have a sense that they are “doing” math or computer science, because school math tends to be directed towards finding right answers to known problems, and computer science is understood to be some kind of drudgery one does in front of a computer, like writing programs or fixing the system.

Mike tells a story of the first time he made a sorting network with children. He showed the children the topic he had considered that day with his university students. The parallel sorting network is a well-understood area of algorithms, but is seldom taught before senior levels of university. Yet students as young as 5 years old can fairly easily understand how to use one, and more significantly, because of the experience they can begin to understand the sort of ideas that computer scientists work with. By “failing” to conceptualize a child's mind as miniature and mildly incompetent, activities like the sorting network invert the notion of age-level hierarchies for mathematical topics, and instead give children of any age and development a chance to engage with an idea in computer science with whatever intellectual horsepower they have.

Since Mike's first experiment with it in the late 1980s, the sorting network activity has been used in many classes—spray painted on grass, chalked on pavement, taped on carpet, glued onto portable tarpaulins, paved in a garden, drawn as miniaturized Japanese board-game versions for schools where space is limited, and crafted in a virtual world for use by students with mobility impairments. It has been done with up to 50 people at once, by Girl Scouts, by senior citizens, by music students, and by guests at birthday parties and a wedding. A number of videos of groups doing it can be found on YouTube. Figure 2 shows some of these variations around the world, and in a virtual world. Figure 2(d) shows the difficulty of using it with students in wheelchairs, motivating the virtual version

in Figure 2(e); Figure 2(f) shows a student who can't walk, yet is "running" around a virtual sorting network. Figure 2(g) shows the idea being implemented in a popular introductory programming environment — in this case it has come full circle, and school children are actually able to implement a kind of sorting network in a program (as long as no-one tells them that it is supposed to be difficult—to them it is just a simulation of something they have done physically). Recently the sorting network has been used regularly to teach music theory, comparing things like note pitches (Figure 2(i)) and note lengths [2]. In Lisa Whittle's classroom, the sorting network was kept on the classroom floor for use in many subjects: ordering distances from planets to the sun (science), molecular weights or densities (chemistry), fractions (math), notes and scales (music), eras or events (history), or priorities (social studies).

The parallel sorting network teaches much more than logic or algorithms. For example, students often try to get to the end as quickly as possible, leaving behind another student who is waiting on the outcome of a comparison. At this point they realize that their haste has caused the whole team to fail, and this is a salutary lesson for those who might be interested in computers but not so good at team work. Eventually it is understood that nobody wins alone; all win together as the sorting resolves the different values into a clear order. The sorting network is a model of cooperative learning, more of a dance or a series of conversations than a race. It is useful to ask people to greet their partner when they arrive at a node, which has led to pleasant surprises as children in some countries salute or charmingly bow in greeting. Another topic of discussion is what sorts of activities can/cannot be done in parallel—digging a hole? digging a trench? getting parcels from Darwin to Christchurch?

The nice thing about this demonstration, and many of the others that have become part of the Unplugged canon, is that children don't want to stop "playing." Tim recalls being asked by a girl for a copy of the network so she could use it at her birthday party, and Frances reports having a class that refused to stop, as they went on and on sorting everything they could, including replacing library books on the shelves [3].

3 Early Activism on Computer Science and Math Education

The sorting network is one of dozens of engaging, thought-provoking activities that could be developed for young students, yet the education establishment found little use for them beyond "enrichment," or novelties offered as a break from rote learning of the calcified topics one is sure to find on standardized tests. Mike expressed his sense of injustice to children at this state of affairs in his dissident 1991 paper about the way math and computing were approached in schools: "Computer SCIENCE and Mathematics in the Elementary Schools" [4]. This paper is essentially a manifesto for the work that continues to this day, and makes the following key points:

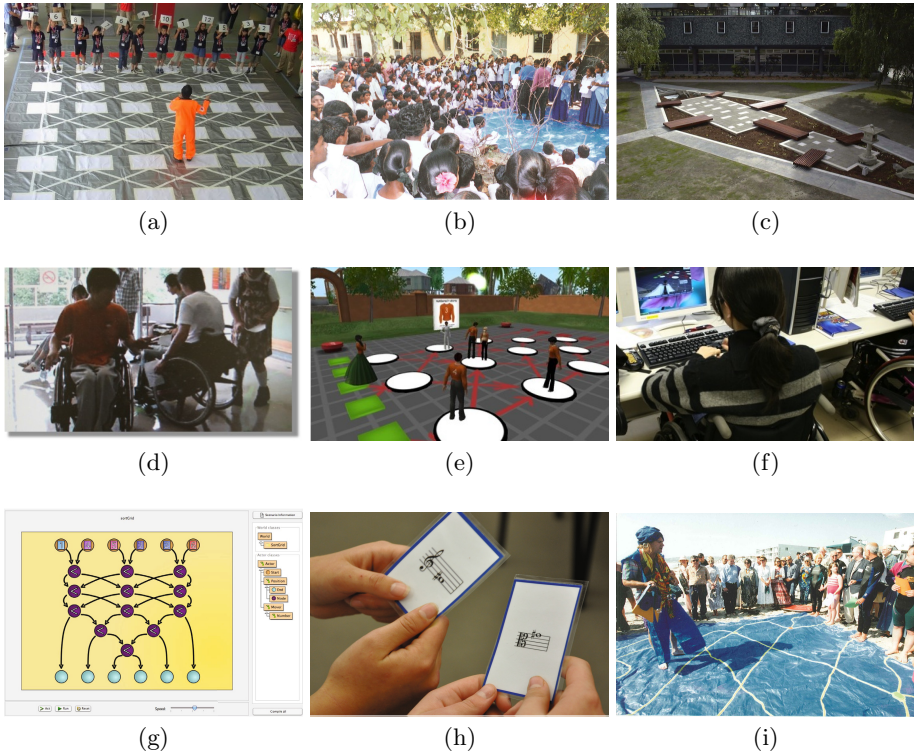


Fig. 2. The ubiquitous sorting network (a) a 12-way network at a kids’ event in Tokyo (b) in India (c) on an “island” of the “seven bridges” garden at the University of Canterbury (d) students in wheelchairs in Japan (e) in Second Life (f) using a Second Life sorting network (g) an exercise in Greenfoot (h) in a music theory class, and (i) at Mike and Fran’s wedding!

- Elementary school students deserve to experience profound and imaginative mathematical ideas. Such ideas shouldn’t be reserved for graduate students.
- Open unsolved problems are the creative drivers for mathematical activity, but children are “taught” a version of mathematics based almost entirely on correct answers.
- Mathematics itself is an “interdisciplinary powerhouse.” The pursuit of mathematical ideas will open doorways and raise interesting questions in the sciences and humanities.
- Mathematics popularization is a research area of basic interest. Exciting mathematical ideas will not find their way to children and their teachers without an effort on the part of mathematicians to communicate about them in accessible ways.

Mike was undaunted by the education establishment’s lukewarm reception to his manifesto and remained bent on drumming up enthusiasm for the work rather than getting bogged down in the politics of curriculum development.

For example, as chair of STOC in Victoria, he created an “Ad Hoc Committee for SIGACTion on Elementary and Secondary Mathematics and Computer Science Education Reform” to coordinate the potential contributions of SIGACT to mathematics and computer science education reform (posted to *theorynet*, March 17, 1992). Members of the committee included: Mike Fellows, STOC 92 Chair; Maria Klawe, Chair, CS Dept UBC; Joe Rosenstein, DIMACS Education Coordinator; Jeff Vitter, Vice Chair SIGACT; Donna Baglio, ACM Headquarters; Avi Wigderson, STOC 92 Program Chair; Vance Faber, Los Alamos; Bonnie Yantis, Los Alamos; Eric Manning, Dean of Engineering, UVic; and Neal Koblitz, Math Dept, Univ Washington. The committee set up a specific project to compile a compendium of algorithmic topics and possible presentation strategies and supporting discussions for the use of interested educators. Rather than issuing a simple bureaucratic plea for material, Mike convinced the UVic Dean of Engineering to contribute \$250 to purchase a raffle prize so attendees at the STOC 93 banquet could earn a raffle ticket by contributing a topic or presentation idea.

The following year, Mike gave a public lecture organized by UVic professor Bill Wadge entitled “Mathematics Education: The Paranoid Theory.” It grew out of numerous rebuffs from academics in mathematics education that implied that he was out of touch with what children needed to learn in school mathematics. The abstract for that lecture reads: “Mathematics, the main engine of modern science, and the widely despised arbiter of social opportunity, functions in many ways like the medieval Church. The talk will focus on the culture of mathematics, the hidden agendas served by mathematics education, and on various associated amusing stories that serve to illuminate the issues, or perhaps only explain the speaker’s evident paranoia. More of a performance than a lecture—whatever happens is at least the work of a mathematician.”

The abstract is accompanied by a Calvin and Hobbs cartoon:

Calvin: You know, I don’t think math is a science. I think it’s a religion.

Hobbs: A religion?

Calvin: Yeah. All these equations are like miracles. You take two numbers and when you add them, they magically become one new number! No one can say *how* it happens. You either believe it or you don’t. This whole book is full of things that have to be accepted on faith. It’s a religion.

Hobbs: And in the public schools no less. Call a lawyer.

Calvin: As a math atheist, I should be excused from this.

The paranoid theory proposed that mathematics (taught as only arithmetic) was used by society as an arbiter of social opportunity, and was really about power, authority and control. He wrote about the Paranoid Theory in “Computer SCIENCE in the Elementary School” [4].

Mike says that he is not sure how much he actually believed in his theory. Almost everyone he has told it to endorses it to some extent, sometimes enthusiastically (the public lecture drew an overflowing audience), yet there is an

incredible resilience to the shopkeeper way of teaching. Perhaps this is because most people have never seen anything else.

The Inaugural Lecture by University of Sydney anthropologist Linda Connor in October 2011 gave us an anthropological perspective on the Paranoid Theory. The core mythology being promulgated in the 18th century had to do with profit and consumerism, and therefore numbers and operations on numbers were of foremost importance. Mike suggested considering the modern “number” to be “networks of relationships,” as in gene regularity networks. One can speculate on what society would be like if children’s math began with networks, continued with patterns, dynamics, processing, and strings of symbols over a two letter alphabet, and eventually came to counting. We plan to explore these possibilities with First Peoples (Indigenous Aboriginal), for whom networks of familial relationships form a precise, intricate and primal construct, at a future workshop.

Mike made enormous efforts to engage people with this view of math teaching, sharing his ideas with any students, teachers, academics or administrators who would listen. He has sought funding and moral support, and worked passionately to help anyone who is interested. Ideas in mathematics and computer science are exciting, and Mike has never been picky about who he shares that excitement with. It might be a researcher of his caliber, or it might be a 5-year-old. Rod Downey recalls feeling pretty bruised by trying to push for improvements in math education, and advised Mike that it was really not worth it as it sucked all the energy out of you. Mike’s boundless energy was such that he persisted despite setbacks, and it’s a testimony to his persistence that 20 years later some of his work has found a foothold in formal curricula around the world.

We now look at the history of Computer Science Unplugged and associated projects, how it began with the *MEGA-Math* project and steadily grew—because exciting ideas won’t be squelched.

4 *This Is MEGA-Mathematics!* —The *MEGA-Math* Project

The impact of Computer Science Unplugged is extraordinary, winning awards worldwide, being translated into many languages, and having its own YouTube channel with sound tracks and subtitles in multiple languages. This section describes the roots of Unplugged, which are found in the *MEGA-Math* project, starting with Mike’s classroom visits to his children’s elementary school in Moscow, Idaho.

The United States in the 1980s considered itself a “Nation at Risk” due to dismal results from international assessments of mathematics education, and to immense differences in achievement traced to race, ethnicity, poverty, and gender, and to huge gaps between low- and high- achievers. One response was to set national standards for school mathematics, an unprecedented venture for the United States, and which resulted in classrooms largely being turned

into test-preparation centers. Innovations tended towards (often artificial) “uses” of mathematics. The result was a collective popular conception that mathematics is incomprehensible, accessible only to a gifted elite, yet very important.

An early approach to putting excitement into math was “Family Math,” which came out of the “Math for Girls” program (created by Diane Resek, Nancy Kreinberg and Rita Liff Levinson) and the “Equals Teacher Training” program, in which engaging, hands-on math activities were developed at the Lawrence Hall of Science (LHS) in Berkeley in the early 1970s. These activities were also a key element of the “Expanding Your Horizons” conferences created in those early days by the Math/Science Network (co-directed then by Lenore Blum and Nancy Kreinberg) and still going strong today. One of the developers of Family Math was Virginia Thompson, and Mike had heard her speak at a meeting in Los Angeles and was influenced by her¹.

In this context, the chance collaboration that resulted in the *MEGA-Math* project began when Mike and Nancy Casey met in 1989 while Mike was at the University of Idaho. Their children were the same ages — 5 and 6. Mike had already developed a few games and puzzles to expose his children and their schoolmates to current research ideas. As a K-12 Language Arts teacher, Nancy was interested in the way children learned reading and writing skills inside a language-rich environment which situates those skills in a context that includes storytelling, fact-collecting, art, music and movement. Although she knew many teachers confident in their ability to organize a classroom in a way that brought children’s active language-learning faculties to bear on the expansion of an array of communication skills that included reading and writing, there didn’t seem to be any teachers with a similar creative vision for mathematics learning. She wanted to know what belonged in that vacuum [5]. At the same time, Mike saw a disconnect between what he did as a research scientist, and what his children did most school days in the name of math.

Mike and Nancy began exploring the possibility of teaching mathematics using the “Whole Language” (also known as “Natural Learning”) philosophy. Children were given large blocks of time to explore, discuss, and write their ideas in notebooks. They began to explore the possibilities of teaching modern mathematical ideas through communication about games and puzzles, having students express their mathematical thoughts in language, a curricular area where teachers felt most secure.

Second grade teacher Prudy Heimsch joined the experiment. According to Mike, every parent in Moscow wanted Prudy to be their child’s teacher. She had the patience to allow the excitement of learning (which may look like chaos) to follow its natural course (rather than provide overly strong guidance for students), and this has become a hallmark of the *MEGA-Math* and Unplugged projects.

¹ Intriguingly, this inspiration ended up coming full circle, as Lenore ended up engaging with CS Unplugged some two decades later after it had traveled from the US to New Zealand and back again!

Prudy's class explored a DOMINATING SET problem², described as "Where should we place a minimum number of fire stations, so that every house is located either on the same vertex as a station, or within one block of a station?", given a map such as the one in Figure 3. Nancy has described the ensuing frenetic and alternately careful, quiet struggles of the children to invent vocabulary with which to explain their thinking about this problem to each other, and write about it [6].

The students were shocked to find out that a teacher wasn't going to give them a correct answer if they waited long enough, and were empowered as they discovered that they didn't need a teacher as they could evaluate one another's fire station maps. This was an inversion of the traditional mathematics lesson where students are handed a series of problems to practice on after being led through a series of strategies for solving them. Also in these sessions, it became apparent that neither Nancy nor the classroom teacher grasped the full richness of the children's problems-solving engagement until afterward when they sat down together to discuss what occurred and plan for the next day. Building these debriefing sessions into the activities was essential to establishing why and how such unbridled classroom enthusiasm was purposeful mathematical activity.

Allowing the students to explore problems in their own way is something that doesn't always come naturally to teachers and visiting university faculty [7]. Yet it is remarkably effective. The students end up taking ownership of the problem without realizing how "hard" it is, and they build confidence that they can cope with novel mathematical situations.

In the workbook that grew out of these experiences, the introduction assures the reader that "Good mathematics ultimately comes from and returns to good stories" [8]. For example, in "Gertrude, Superperson and the Monster Recover from a Disaster," the stage is set for a discussion that may well end up with the 4-color map theorem, although the phrase "map coloring" never appears in the story, which begins:

HORRORS! The Land of Many Ponds has been clearcut! Where there were once tall, tall trees and flyways, there is nothing but stumps and empty space.

Gertrude, Superperson and the Monster meet in one of the ponds to talk over what they should do.

Gertrude swam round and round in furious circles. "Oh we are doomed, just doomed!" she wailed. "As soon as it rains, WHOOSH! the soil will begin washing away. . . The (pond) water will get silty and soon the ponds will dry up. Oh it's awful! We are doomed!"

² A slight variation of the story uses the VERTEX COVER problem, where every *edge* in the graph is incident on a vertex in the solution set; in contrast, the DOMINATING SET problem requires that every *vertex* in the graph is adjacent to a vertex in the solution set. The difference in the story is whether the houses are on the roads or on the intersections. Later versions of the story included choosing the placement of ice cream stands and the location of wells.

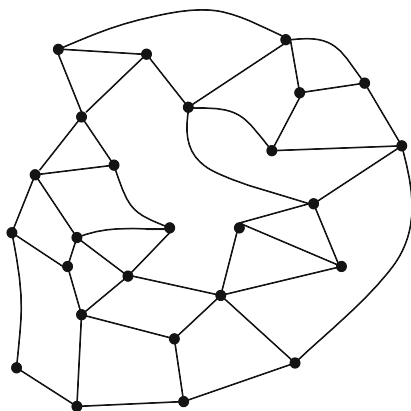


Fig. 3. A graph used as a map of the town that needs the location of fire stations to be decided

The characters set off to make plantings to prevent erosion and decide to plant colored flowers in a way that makes the old boundaries still clear. In the process they are likely to encounter ideas relating to the four-color theorem, the exponential complexity of exhaustive searches, and special cases of maps that require fewer than four colours.

Examining a dots and lines network of relationships requires a “spatial” reasoning that seems quite different from that used in calculating addition or subtraction, with the result that some students who always were the “best” in math no longer knew what to do, while those who normally did not do well became leaders. Mike has described one little boy who had a reputation as the class trouble-maker. He became so excited working on the network that he exclaimed, “This is math? This is MEGA-mathematics!!” and thus was named the project. They also found that *teachers’* confidence levels with mathematics skyrocketed when they were encouraged to consider how their students expressed and developed their mathematical thoughts in language, the area of the curriculum where they were often most secure.

In his 1991 paper, *Computer SCIENCE and Mathematics in the Elementary Schools* (the most definitive description of the vision and innovation of the project), Mike urged that children be respected as genuine (child-sized) researchers (the paper was published in 1991 on the *MEGA-Math* site, and later appeared as [4]). Mike states: “In the same way that children’s art is interesting as art and children’s writing is interesting as writing, mathematics with children can be interesting as mathematics.” Research problems sometimes “turned up” during classroom visits. For example, Jan Kratochvil visited for several months from Prague, and he and Mike noticed that if two children take turns coloring in the regions of a map, the Four Color Theorem assures that four colors are enough, provided you are coloring perfectly and strategically. But how many colors will it take if one player is a child who plays legally but not strategically?

Or, if they are trying to “force” the map to require more than four colors? (An upper bound was established by Kierstead and Trotter in 1992; they show that 33 colours suffice for a planar graph, and they establish 8 as a lower bound [9].) Children are always challenging the rules of the game, and they don’t know which questions they shouldn’t ask.

In the 1991 paper, Mike describes (retrospectively, he says) the goals of his classroom visits:

- To show that mathematics is fun and full of stories, activity, invention and play.
- To show that mathematics, like dinosaurs and outer space, is a live science with visible frontiers of knowledge.
- To present the essential unity of mathematics and computer science and display the intellectual core of the latter.

These early experiences led to a lively discussion of how math should really be taught in schools. In that discussion, juxtaposed to these experiences of very young students having an exciting time exploring graph theory we have school curricula that teach the ingredients of math — operators, fractions, probability and so on. Yet if these ideas are the entire curriculum, students are merely learning ancient techniques from what appears to them to be a dead discipline [5].

4.1 The *MEGA-Math* Book

The ideas that led to the *MEGA-Math* project were gradually collected, and built up a grass-roots level of interest. In 1993, under the leadership of Vance Faber and Bonnie Yantis of the Computer Applications and Research Group at Los Alamos National Laboratory, Mike became principal investigator on a grant entitled *Research On Mega-Math: Discrete Mathematics And Computer Science for Children*, which resulted in the publication of the activities in a 134-page, *This is MEGA-Mathematics!* workbook by Casey and Fellows [8].

The *MEGA-Math* workbook covers six topics: map coloring, knot theory, graph theory, finite state machines, algorithms, and infinity. The introduction states: “We hope that these materials will provide opportunities for children and their teachers to experience mathematics in ways it is experienced by mathematicians and scientists. Mathematics is lively and exciting; it is a field more akin to art and poetry than many people think.” Written at a level an enterprising student could grasp, it contains descriptions, explanations, games, stories, pictures, problems and questions for discussion. It assumes that teachers are not familiar with the topics. Detailed information describes how the activities can be used to meet the curriculum goals outlined in the *Standards* of the National Council of Teachers of Mathematics (NCTM) (standards.nctm.org). In their article, *Implementing the Standards: Let’s Focus on the First Four* [10], Mike and Nancy have argued that in order to properly address the NCTM elementary school standards — reasoning, problem-solving, communications, and connections — *new content* must be introduced into the K–4 curriculum. The authors show by

example how the goals of the standards can be achieved using material from computer science and discrete mathematics, and they describe their approach to teaching mathematics as parallel to the “Whole Language” approach to teaching reading.

After the publication of *This is MEGA-Mathematics!*, a larger following developed. In 1994, with continued funding from Los Alamos, the materials were expanded and put up on the then-new World Wide Web [11]. *MEGA-Math* material was soon used for summer camps, and in 1994, a presentation of these materials won first prize for “New presentation ideas” in a Canadian summer camp national organization. Activities from *MEGA-Math* were adapted by other organizations, including DIMACS (Rutgers University, NJ), Family Math (Lawrence Hall of Science) and the Math Department, University of Illinois at Chicago. It ultimately had some influence on curriculum design, especially in British Columbia, Montana, California and New Jersey.

By the time *MEGA-Math* was taking off, Mike had moved from the Laboratory of Applied Logic in the Department of Computer Science at the University of Idaho to the Department of Computer Science at the University of Victoria, British Columbia. Mike continued to develop games and activities, trying them out in elementary schools in Victoria while Nancy collaborated with several teachers in Moscow, Idaho doing week-long sessions (including teacher-debriefing) like those which had worked so well with the MINIMUM DOMINATING SET problem. By 1995, over 1200 copies of the free workbook had been distributed to more than 400 individuals, including bulk orders to organizations such as DIMACS and Family Math program at Lawrence Hall of Science. It was used in departments pursuing mathematics education reform, demonstrated in many classrooms, and parts had been translated into Spanish [11].

MEGA-Math took a radical approach to engaging very young students (including pre-schoolers) with what is conventionally regarded as post-high school topics, such as graph theory and finite state machines. New pedagogical notions were introduced such as linking the concepts with playful stories about monsters, animals and bakers, and representing a math problem physically, with giant ropes for knot theory or large tarpaulins with networks constructed on them made of colored tape. Significantly, it established that there is value in the head-first, constructivist approach, where students are given a hard problem (such as finding a dominating set), and are left to explore it in their own way. The innovative *MEGA-Math* approach set the scene for expressing computer science so that elementary school students could engage with the deep concepts that get computer scientists excited. Furthermore, due to Mike’s infectious enthusiasm for the possibilities expressed in *MEGA-Math* and eventually Computer Science Unplugged, many in the parameterized complexity community have joined in presenting innovative activities in their children’s classrooms and other public venues, opening up public understanding and participation in mathematical science.

MEGA-Math continues to have a life of its own — in 2011 the web site was getting fairly constant traffic at around 4,000 hits a month, and the ideas

are referenced in popular lists of math and computing activities. But one of its biggest impacts is that it laid the groundwork for the soon-to-be-created Computer Science Unplugged project.

5 Computer Science Unplugged—Genesis

The Computer Science Unplugged project began through a chance online meeting of Mike Fellows and Tim Bell in March 1992 via an Internet newsgroup. From about 1989 Tim had developed computer science material for a “Science Extravaganza” in Christchurch, NZ [12]. The extravaganza was a temporary science exhibition set up annually in the late 1980s, reminiscent of the San Francisco Exploratorium, and in 1991 evolved into a permanent science center called “Science Alive!” At the time computer exhibits in science centers didn’t demonstrate “real” computer science, and Tim was interested in preparing engaging material that communicated the great ideas of computer science, and not just programming or using a computer. Amongst other things he had developed a demonstration of tractability where you have to watch a program count down while it solves the Travelling Salesperson Problem [13] — which can be dressed up as the “birthday party problem,” dropping n kids home after a birthday party. At the exhibit the child chooses n , and the program animates an evaluation of all $(n - 1)!/2$ possible paths, one per second. For a few cities, this is very fast, but for $n = 27$, the system brings up a timer that counts down from 6,394,144,170,576,570,000 years, one second at a time. The ensuing humor emphasizes the futility of exponential time algorithms, and provide opportunities for follow-up questions (e.g. what if the computer was a million million times faster? Then it would only need 6,394,144 years!)

During 1992 Mike and Tim exchanged ideas for communicating computer science to young children. In August 1992 Tim was asked to give a talk to his son Michael’s class at Shirley Primary School, Christchurch (just two months after Michael started school). It was part of a series of talks by parents about their jobs, and followed on from other parents such as policeman who brought a police car for the kids to see and a nurse with real bandages for the kids to try. Faced with the challenge of trying to communicate how one could make a living working with algorithms, and inspired by discussions with Mike, Tim decided not to bring a computer at all — after all, how convincing would it be to show 5-year-olds a computer running a slow and then a fast algorithm? This led to the development of material such as the cards for teaching binary numbers, and the parity card trick. The session turned out to be a resounding success, and armed with these ideas and others gleaned from Mike, by October 1992 Tim was making regular visits to Shirley Primary school, and had engaged illustrators (initially Gail Williams and Malcolm Robinson) to design handouts that would appeal to the students. Two senior computer science students (Gwenda Benseman and Richard Lynders) also helped with the visits.

It turned out that being able to teach binary numbers using 5 cards drew a lot of interest. Teachers and parents who observed it over the years often commented

that they felt empowered because they understood this mysterious idea of binary numbers, bits and bytes, which previously seemed to be a secret language of the in-crowd. It has been popular as a demonstration to senior citizens for the same reason, once again giving life to Mike's approach of not paying attention to boundaries and allowing anyone with curiosity to explore ideas from math and computer science.

As the collaboration with Mike developed, Tim organised a one-month visit to Victoria BC in March 1993, funded by a University of Canterbury Erskine fellowship. *This is MEGA-Mathematics!* had been published by that time, Mike had been working on "Kid Krypto" [14], and he had considerable experience in schools using these ideas to communicate with elementary aged students.

During that time Tim gained experience observing events that Mike ran. Mike's somewhat organic approach provided a whole new perspective on communicating with students. Often ideas for presenting concepts would be developed the night before, or in the car on the way to a school, or even in the classroom as the students explored the concepts. Tim remembers driving up Vancouver Island to Hans Helgeson school one morning, leaving town somewhat less than 45 minutes before the class started (the drive was about 45 minutes). As Tim was busy calculating the expected time of arrival, Mike spotted a gas station with a store, pulled in, and like a child in a candy store went around collecting items like cans of spray paint, string, and coloured tape. They arrived at the class a few minutes after the appointed time, but oblivious to any concern from teachers, and with no lesson plan at all, Mike launched into a session with the students and soon had them eating out of his hand as he spun stories about pirates or lost animals, whose fictional problems could be solved using only a finite state automaton or sorting network. Grass was spray painted, tangles of string and tape emerged, and suddenly it became obvious that students were gaining a deep understanding of advanced ideas from computer science and math.

Of course, at such sessions some teachers were probably unsuccessfully trying to work out which curriculum boxes they could tick after the session — it couldn't be math because there was no arithmetic, and it couldn't be computing because there were no keyboards! Even the students may not be aware — on a different occasion a student commented "I'm glad you're here today, otherwise we'd have to do math"! On that occasion, instead of "math" the student ended up doing things like combinatorial problems and modulo 2 check sums.

An important observation about this "organic" approach to teaching is that although it appeared to be chaotic as students threw themselves at problems with some taking obviously sub-optimal approaches or exploring beyond the boundaries given, Mike's infectious enthusiasm and gripping story telling had them engaged and fully aware of the context he had given them. Later it became clear that this student-driven exploration was an approximation to constructivism, a well-known teaching philosophy that came to be a valuable approach to successful outreach programs. Instead of telling students some information and leaving them to digest it or be impressed that the speaker knows a lot, the students

explore the problem for themselves, finding good or bad solutions, and enjoying the journey rather than worrying about the destination (which after all is only a meaningless solution to an artificial problem).

Tim's 1993 visit to Victoria resulted in the drafting of what became the Computer Science Unplugged "original" edition, although because of various delays and distractions, it was a couple of years before the book became available.

5.1 Mathmania

Also in 1993, at Victoria BC, Mike began the "Mathmania Society"³. One of the Mathmania group's first events was a "Mathmania in the Park" day, which was on Saturday 13 March 1993, during Tim's visit to Victoria. Figure 4 shows the poster advertising the event, and Figure 5 shows snapshots of the event, with families doing serious math as part of a picnic in a park. The event included a "finite state treasure hunt", which was a pre-cursor of the Treasure Island game: kids would go to a "station" in the treasure hunt (an adult with a sticky label) and ask to take the "A-train" or "B-train", which would result in them being directed to run to another station. The knots made out of inch-thick rope provided a tactile experience for even the youngest attendees to explore the basic elements of knot theory!

The mission statement of the Mathmania Newsletter states that it is "introducing mathematics as it is done by professionals." The first newsletter (dated 05/94, presumably May 1994) says "We hope we have given a hint of how this can be done in a way that is engaging, accessible, and that does not require the teacher to invest a great deal of time or funds. Hopefully, the activities allow for the playful, open-ended, provoking and enjoyable style of mathematics enjoyed by mathematics researchers everywhere."

The newsletter quotes some key principles from a paper written by Mike:

- There is an essential unity of mathematics and computer science.
- The competencies required for the increasingly computerized world are essentially mathematical. It is a serious (and common) mistake to make a fetish of the machines.
- The intellectual core of computer science can be presented to children even in situations where there are no computers (for example, in countries or school systems that cannot afford them), laying a foundation for later computer science education. Many of the core ideas of computer science are best introduced without machines.
- Computer science represents a tremendous flowering of mathematics. It is particularly good news for children because it is a treasury of accessible, colourful and active mathematics.

³ Not to be confused with a group with the similar name of "MathmaniaCS" in Urbana-Champaign, Illinois, who do excellent related work combining the *MEGA-Math* and Unplugged material as well as other original material.

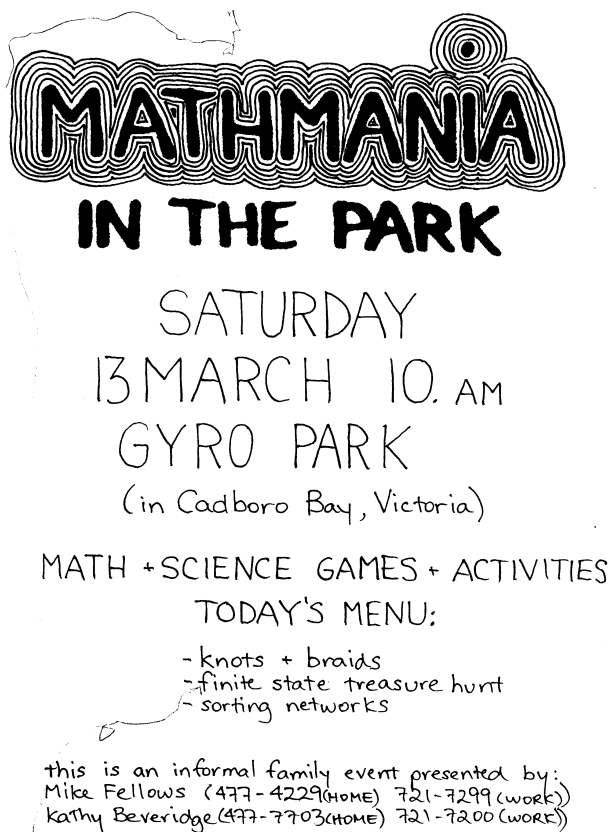


Fig. 4. Poster advertising the “Mathmania in the park” event in 1993



Fig. 5. Mathmania in the park (a) A “station” in a Finite State Machine giving directions (b) a sorting network spray painted on the grass (c) knot theory for the whole family

- Most children in grades 1–4 are never exposed to mathematics. Arithmetic is not mathematics!
- Most children in these grades are never exposed to computer science, despite all the PCs in the classroom. Programming is not computer science!

Later the Mathmania organization became more formal, and in 1995-1997, while at UVic, Mike secured funding to found the “Mathmania Society for Public Education and Appreciation of Mathematical Sciences,” incorporated under the Society Act by the Victoria, BC Registrar of Companies. Mike was a Director of the Society (along with Nancy Casey, Day Kirby, Gerald McLean, Kathy Beveridge and David Vogt). A joint project of Mathmania and the Canadian Mathematics Society was to create a website of kid-sized open mathematics problems with prizes under a project called “Erdős for Kids.”

5.2 The “Original” Unplugged Book

In parallel to the development of the Mathmania society, the book by Mike and Tim about doing computer science without computers was drafted. General principles and an outline were drafted during Tim’s 1993 visit, and it was developed over the next couple of years. Until 1995 its working title was “Junior Algorithmics”, basing the title on that of Harel’s 1987 book “Algorithmics: the spirit of computing” [15], which was a popular account of computer science written for an adult audience. The full working title was: “Junior Algorithmics: Computer Science for kids (and grown-ups who don’t mind having fun)”. However, early on it became apparent that the word “Algorithmics” was daunting or confusing for many people.

The name “Computer Science Unplugged” was coined around 1996. The term “Unplugged” came from a style of music that had become well-known around that era. During the 1980s it had become popular for artists to perform versions of their music “acoustically,” especially using acoustic guitars instead of electric guitars with effects. The term “unplugged” for such music seems to have been used first for the “MTV Unplugged” series that started in November 1989. The Unplugged format became particularly popular with a 1991 Paul McCartney recording, and then Eric Clapton’s 1992 “Unplugged” album, which received many awards. Thus at the time the Computer Science Unplugged material was being developed, the term “Unplugged” had just become well established as being associated with avoiding having music cluttered by technology, returning to the essence of the music. This resonated strongly with the work that Mike and Tim were doing, trying to avoid the distraction of the technology so that students could appreciate what is really happening.

The term “Unplugged” has come to have strong recognition in the computer science education fraternity and is often used without definition to describe some aspect of a teaching or outreach program. It is often associated with kinesthetic activities, although when used properly it also has much visual and oral learning. At least one independently written book, “Algorithms Unplugged”, has picked up the term and the spirit of the approach [16].

Unfortunately the word has posed considerable problems for translators, since the translation in other languages generally has pejorative meanings that imply that it is describing something that has no power, is not working, or is broken. Taking the lead from the music industry, most translations either keep the English term “Unplugged”, or replace it with a more inspiring description of the material. For example, the Korean subtitle means “Learning Computer Science with Games”.

The idea of rejecting the use of computers has created some soul-searching moments for the authors, particularly when it became clear that the material needed to be made available online, and then supporting material such as videos and online games started to appear. More recently “Unplugged” has even been implemented in virtual worlds! However, an analogous situation occurs with unplugged music—every “Unplugged” recording uses electronic devices, often including guitar pickups and even electric organs. In fact, the cover of the best-selling Unplugged album by Eric Clapton clearly shows a microphone that is plugged in! Unplugged is really an attitude rather than a technique. The pragmatics of making the material widely accessible mean that we don’t eschew computers per se, but we do avoid the situation where the physical device becomes the object of attention and displaces the great ideas that will engage students’ minds. Of course, right from the start computers were used to design and communicate the Unplugged ideas, although it was tempting to make it a hand-written book.

An ongoing debate about the title has also been the use of the term “Computer Science”; this term is very well established and is reflected in the names of many university departments that teach the subject, but it can create confusion for those not familiar with the field (for example, is it about using computers for science?) There are competing terms such as the European “Informatik,” and the general term “computing”, but since the main international professional organization, the ACM, uses the term “computer science,” that phrase was chosen. The abbreviation “CS” is almost more effective because it removes emphasis on the two words that cause the confusion, and the project is now often referred to as “CS Unplugged.” Another minor distraction with the term “unplugged” when associated with computers is that it is also used to refer to wireless devices—for example, computingunplugged.org is a magazine about mobile gadgets. The use of the term “computer science” (rather than “computing” or “computers”) generally avoids this confusion, or at least, it gives an opportunity to point out that computer science has a very particular meaning.

Progress on the CS Unplugged book was slow for the first couple of years as during this period the project was still a side interest for both Mike and Tim—Mike’s early work in parameterized complexity was keeping him busy, and Tim had become involved as an expert witness for several major compression court cases (in fact, he was first approached to help Microsoft while in Victoria on the March 1993 visit). Requests and encouragement from colleagues to see the material completed gave a sense of urgency, and in 1995 Ian Witten was invited to become involved to help massage the collection of disparate ideas

into a coherent collection that would be useful for people involved in outreach. Ian also proposed and added the sections on Artificial Intelligence and Human-Computer Interaction, since these were areas he had an interest in.

The coverage of topics wasn't entirely haphazard. There was an effort early on to systematically list computer science topics so that fairly broad coverage would be possible. Every one of the five dozen chapter titles in Dewdney's book "The Turing Omnibus" [17] was considered for a possible Unplugged topic, and the ACM curriculum was checked to see if the coverage was representative. There were some topics that weren't covered initially because of the criterion that the activities had to be engaging for children, and it wasn't always possible to come up with a relevant one. Also, given the authors' interest, there is some bias towards algorithms, compression and tractability!

The activity on text compression was invented by Michael Bell around 1994. He was writing up a diary and had decided that instead of writing out every word, he would just put arrows pointing back to where he had already used the word before to save writing. Tim observed this, and was about to reprimand his 6-year-old son for being lazy when he realized that it was a form of Ziv-Lempel coding. To add irony, at that time Tim was testifying in a US\$339 million court case that a patent for a similar form of Ziv-Lempel coding should be regarded as invalid!

While the book was being written, the material was tested extensively in schools, mainly those where Mike and Tim's children attended (Shirley Primary School in Christchurch, and South Park School in Victoria BC); and also schools close to the respective universities (Ilam School in Christchurch, and Hobbes Elementary School in Victoria, BC). Often apparently interesting ideas didn't work as well as expected, and vice versa. An important principle was not to publish something that hadn't been successful, and the activities contain advice on things that can go wrong and ways to avoid (or embrace!) such problems. Sometimes suggestions for improvements came directly from teachers and children, such as using a balance scale for sorting. The trials were also run by university students who did them as course projects or just out of interest; those who helped included Gwenda Benseman, Richard Lynders, Sumant Muruges and Matt Powell.

A rough version was complete by 1996, and parts of it were distributed informally (including 100 copies in the Coquitlam BC school district) while a suitable place to publish it was found. The authors were aware that teachers were more likely to take a book seriously if it was published by a well-known educational publisher rather than distributed as photocopied notes or on disk. Colleagues had expressed great enthusiasm about the 1996 drafts, and so it came as quite a surprise when it was rejected when sent to a publisher. In fact, between 1996 and 1997 it ended up being submitted to 27 publishers, and not one accepted it. The general response from the publishers was very positive about the book contents, but they couldn't pigeon-hole it in their range of offerings! One publisher wrote that they "will not pursue the idea of publishing the book" yet described it as "your wonderful volume..." and said it "would be a real pity not to have

this book released.” The difficulty was that it was breaking new ground. A children’s publisher said it “may be too academic for children”, while an academic publisher referred us to a children’s publisher! An education department of a publisher referred it to their computing department, but then the computing department said that they couldn’t publish a book if it wasn’t about how to do things on a computer!

While these frustrating conversations continued, the authors completed the book, and when it became clear that no-one would publish it, the authors decided to sell it as “shareware” online, from about March 1998. Samples were available online at www.unplugged.canterbury.ac.nz, and interested people could email or fax in a US\$15 credit card order to get an electronic copy (download or CD-ROM). For US\$75 they could get an “institutional license” to make multiple copies in a school or university. Some dozens of copies were sold this way, and many more were given away. There were probably many bootleg copies around too, which the authors were relaxed about since it created an even wider distribution (there was no copy protection on the PDF files distributed), and the only goal of sales was to fund further work on the project.

Free copies were distributed any time funding could be found to make a print run. For example, in October 1998 Tim received a contract through the Royal Society of NZ Science and Technology Promotion Fund, and ran teacher training days in April/May 1999 in Canterbury, Waikato and Auckland, in which teachers got to participate in the activities, and were then given a copy of the book to take away.

Around this time much of the material had also appeared in a slightly different form through the “MathmaniaCS” project at the University of Illinois at Urbana-Champaign (the name “MathmaniaCS” was defined as “persons exhibiting an excessive passion for MATHeMatics and Computer Science”). The project was run by Lenny Pitt, Cinda Heeren, and Tom Magliery; they had written a manual for teachers, as well as providing on-line instructions and a lending library of materials for running Unplugged sessions. They ran various outreach events, including family math nights, camps and classroom visits. Their material is available through www.mathmaniacs.org.

By 1999 the book authored by Mike, Tim and Ian settled into a final version called “Computer Science Unplugged. . . offline activities and games for all ages”, which contains 20 activities. This one is usually referred to as the “original book” [18]. It had been intended to be useful for teachers (the preface says that it is “principally for teachers who would like to give their classes something a bit different from the standard fare, teachers at the elementary, junior high, and high school levels”) but it became apparent that it was too advanced for teachers with a weaker background in math and computer science, even though they found the topics interesting. Teachers who had seen the activities demonstrated readily adopted them, but in the end the book had been written by three academics and didn’t suit the audience, given that almost no school teachers would have a formal background in computer science, and few would even have taken advanced study in math!

5.3 The “Teachers’ Edition” of Unplugged

As it became apparent that a more teacher-friendly version was needed, the authors decided that this would be best achieved by having some teachers re-write the material. In 1999 Tim obtained funding for this from the local Brian Mason Foundation, and hired two teachers, Robyn Adams and Jane McKenzie, over their summer break (December 1999 to January 2000) to do this. Neither had a strong computer science background, but both were experienced teachers with a strong interest in science, and thus were able to write something that would appeal to other teachers who hadn’t done computer science before. They were co-teaching a class at a local primary (elementary) school, and Jane had been using Unplugged material for some time; she had also been a writing tutor in the Canterbury computer science department.

In addition to improving the writing, Matt Powell was engaged to add a lot more cartoon images to make the material more attractive for students. Matt was a computer science student and so was able to create relevant cartoons that embodied a deeper understanding of computer science. The cartoon characters and logos that he created for this remain the main visual “branding” of the Unplugged project.

Because the time for writing the teachers’ book was limited to just the summer, only the first 12 of the 20 “original” activities were completed. It turned out that this was more than enough to satisfy demand; we suspect that most people only use a few activities from Unplugged, and it is more important to have a few approachable ones, with more available in the original form for those who are keen.

The teachers’ edition was released in 2000, again as “shareware”. Minor changes were made over the following years, and the main version referenced is from 2002 [19]. It has a different subtitle that distinguishes it from the “original” version: “Computer Science Unplugged: An enrichment and extension programme for primary-aged children”. The use of New Zealand terminology (“programme” and “primary”) is an acknowledgement that the “translation” was funded by a New Zealand organization even though the largest audience was US-based. The funding also meant that copies were free to Canterbury/Westland teachers, and discounted copies were distributed in New Zealand; at the time the NZ dollar was quite weak, so the shareware fee was fairly nominal for people overseas, and the book had a wide distribution. The on-demand printing service lulu.com was also used to distribute the book, and a large number of copies were sold (mainly in the US) through this service.

The funding for the teachers’ edition was focussed on the Canterbury and Westland regions of New Zealand. As part of the followup to re-writing the book, Tim ran workshops around Canterbury and Westland in 2002. Westland is one of the most isolated parts of New Zealand, and visits from computer scientists were no doubt rare. In late 2002 Tim fulfilled the funding obligations by doing a tour of schools in Westland, in the towns of Greymouth, Hari Hari and Hokitika. One particularly memorable class was a “technology” class where

Tim found himself speaking to a group of 15 year old girls who were studying a particular form of technology — food technology. They had no interest in computers or math, which became obvious very quickly. Fortunately the versatility of Unplugged kicked in — several of the activities involve food (including paying in candies to make comparisons when searching, the divide and conquer cake, and transmitting a chocolate bar securely using an encryption protocol). These generated some interest, but what suddenly got them all engaged was when Tim mentioned that the “From:” field in an email isn’t guaranteed to be accurate, and that emails in plain text can be intercepted and read. Suddenly they were very interested in cryptography—apparently they wanted to be sure that their communications were private and authenticated!

The focus on Unplugged in New Zealand increased because Mike lived in Wellington, NZ, from 1999 to 2001, so by chance all three main authors were living in New Zealand, albeit in three different cities. To add to the confusion, the university that Mike moved to in NZ was Victoria University of Wellington (VUW), not to be confused with Victoria BC! During the time in Wellington Mike taught a summer class in introductory computing with Frances Rosamond, and the Unplugged activities featured heavily in the class. They weren’t just used on campus; the class (mainly adult students) was sometimes run downtown in outdoor locations. The sorting network activity was done outdoors in a weekend next to New Zealand’s national museum, Te Papa.

5.4 The Unplugged Shows

A new format of Unplugged began in 1998. Tim had the opportunity to present Unplugged at the Edinburgh International Science festival in April 1998. The presentation was mainly based on the Unplugged activities, and was originally intended as a classroom style presentation. However, after seeing other shows at the festival, Tim quickly adapted some of the activities to a more theatrical version: the small desktop binary cards became large A4 cards with one held by each child, coloring in small pixels with a pencil became a can of black spray paint putting inch-high pixels on the wall, and the ubiquitous sorting network was laid out as large as possible on the floor using colorful tape.

The goal was to develop something that was a cross between a pantomime, magic show, and science demonstration, with plenty of audience participation (inspired by Mike’s chaotic classes where learning and fun were apparent in abundance). It was also intended to provide a computer science equivalent to the science center demonstrations where a chemist would hammer in a nail with a frozen banana, or explode a can containing custard powder dust (of course, one wonders how much chemistry children learn from these demonstrations).

Observing other popular science shows at Edinburgh made it clear that advertising was important; the most popular events had an attention grabbing photo, used humor, and usually mentioned food; the educational value was assumed! This led to the following advertisement for the shows:

This wacky show takes kids (and the young at heart) through some of the great ideas in computer science, using low-tech games, magic tricks and stories. Come and see the giant fax machine, find out how to feed a crowd and always have food left over, and learn new ways to keep information secret.

Building on the experience at Edinburgh, the show was revised to make it as engaging as possible for a large audience, and was presented in the middle of 1998 at the Christchurch “Kidsfest” mid-winter festival for 5 to 12 year old children, and the Dunedin (NZ) International Science Festival. Matt Powell, who had theatre/comedy experience, assisted with the shows, acting as an uninformed assistant who assumed that a computer science show would need to be all about computers. The story line was Tim demonstrating to him, with the help of the audience, that you can explore great ideas in computer science without any computers at all⁴. Extra ideas were added including the “binary birthday cake” (celebrating an audience member’s birthday with candles coding their age in binary), and ideas from Mike’s work with Neal Koblitz and others on cryptography [20]. Figure 6(a) shows the inevitable sorting network race in a show from 1998.

The Kidsfest shows went remarkably well, and were sold out year after year as new generations of young children came through; it was also performed at the 1999 Australian Science Festival in Canberra. A survey of the show indicated that it had a very positive impact on the audience [21]. Informally, the most gratifying feedback was from parents and grandparents who had brought the children along; they repeatedly reported how empowered they felt because they left with a deep understanding of some key ideas from computing.

The show accidentally found a mascot for Unplugged. “Arnold the Wonder Parrot”, a puppet that squawked when squeezed, was introduced by Matt Powell in the 1998 Kidsfest show, originally as a pun on “Parity Error.” In the surveys about the shows Arnold regularly featured as one of the most popular elements, and so he remained a part of the show. He eventually became the Unplugged mascot, appearing around the world in cameo photos at events. In Figure 6(b) he appears with Jason Alexander, a postgraduate student who ran later Kidsfest shows, and in Figure 6(c) he appears in a photo advertising a 2008 video of the show run by Matt Powell and Javier Jarquin.

Ideas from the shows and Unplugged in general were provided as background to Christopher Bishop (Chief Research Scientist at Microsoft Research Cambridge) as he prepared the televised 2008 Royal Institution Christmas lecture. This was the first “Faraday lecture” in 183 years on computer science.

By the year 2002, 10 years after the Unplugged collaboration began, the main ideas had been published, a show had been developed, and it seemed that this “hobby” could be put aside. Indeed, at that time computer science departments were at what turned out to be the peak of an incredible growth in enrollment,

⁴ Almost exactly ten years later Matt played the role of the informed demonstrator in a video of the show that went viral on YouTube (www.youtube.com/watch?v=VpDDPWVn5-Q).



Fig. 6. Computer Science Unplugged: The Show (a) Matt Powell timing a sorting network race at one of the first CS Unplugged shows, 1998 (b) Jason Alexander with Arnold the Wonder Parrot (c) Matt Powell and Javier Jarqin with Arnold, in a publicity shot for the 2008 version of the show

and given dire shortages of teaching staff, it was hard to motivate faculty to spend time using material like CS Unplugged to drum up more business when those staff were needed to teach the overwhelming number of students that had arrived.

However, in 2003 that was about to change.

6 Computer Science Unplugged—Maturity

Two important changes in 2003 triggered a significant increase in interest in the “CS Unplugged” material. First, it was becoming clear that the overwhelming interest in studying computer science at universities around the turn of the century was dropping off sharply⁵; second, and more significantly for Unplugged, the ACM released a proposed K-12 computer science curriculum [22] that gave 15 examples of ways to teach computer science in schools; five of those examples were direct references to CS Unplugged activities, and another two were from *MEGA-math*. This combination of events triggered increased interest in CS Unplugged from around the world over the next few years. As well as requests for copies of the books and permission to use material, there was a sudden interest in producing translations of the book.

By 2006 the first translation of the teachers’ edition had been produced in Korean [23] by the Computer Education department of Korea University. Because South Korea already had a strong culture of teaching computer science in schools, the book became well-known around the country, and later when Tim visited the university they ran all-day Unplugged events, with a keynote presentation from Tim followed by reports from various educators about how they were using Unplugged.

Meanwhile CS Unplugged was alive and well in Scandinavia. While working with Jan Arne Telle on Parameterized Complexity, Mike had visited Bengt Aspvall’s group at the University of Bergen, Norway, around 1997, which was the

⁵ This has been tracked by the CRA “Taulbee survey”
www.cra.org/resources/taulbee

starting point for Unplugged work in Scandinavia. Bengt has since been giving workshops in Sweden and Norway, and produced a Swedish translation. In 2005 Bengt had just finished six years as pro-vice-chancellor at Blekinge Institute of Technology, Sweden, and was able to visit Christchurch for a couple of weeks, where he and Tim collaborated on developing more Unplugged material. One particularly notable outcome is that they realized that teachers were more likely to use Unplugged if they had seen activities in action, and that it would be good to make some videos to demonstrate this. Because it was December and schools were having a quiet period towards the end of the year, they were able to arrange filming the very next day. The video production was done by Michael Bell, and three videos were completed within a matter of days. YouTube had just been created, so the timing was just right to make distribution easy.

From August to October 2006 Tim was on leave after a long stint as HOD, and visited around 18 institutions in the US, China, Sweden and Canada. Although much of the trip was intended to gather information for a course he was planning on computers and music, it became clear that the universities were much more interested in hearing about Unplugged, and the effect of the trip was to stir up even more interest in the project. During the two months he gave about 16 workshops, shows and seminars specifically on the Unplugged project with audiences from elementary school pupils and teachers to university academics and Google engineers. By visiting schools and enrichment programs where the Unplugged material was being used, and through discussions with computer science lecturers, school teachers and education officials, he was able to come away with valuable feedback and ideas that stimulated a new phase of the Unplugged project.

Another place that Unplugged had gained visibility was through the “Computer Science for High Schools” (CS4HS) program (see www.cs4hs.com). This event for high school teachers started as a pilot for 48 teachers in 2006 at Carnegie Mellon University (CMU), where high school teachers were funded to spend a full weekend on campus to learn about computer science and computational thinking, and get ideas that they could take back to their schools [24]. The event was sponsored by Google, and one of the invited speakers at the pilot event was Craig Nevill-Manning from Google New York. Craig was an ex-student of both Tim Bell and Ian Witten, and so when asked to speak to high school teachers he chose to use some material from CS Unplugged. A survey of teachers at the event reported that the Unplugged workshop was the second most popular out of 11 workshops offered [24], which no doubt was a contributing factor to it being adopted as a regular part of the CS4HS event. Tom Cortina reports that in subsequent workshops Unplugged continued to come in as a highly rated and relevant activity for the teachers — they liked it because they could use it immediately in the classroom. Tom Cortina and Lenore Blum ended up running workshops elsewhere on Unplugged, further spreading interest (for example, Tom ran workshops for several years at NECC). In 2007 the CS4HS program was run at two additional US universities [25], and by 2011 had expanded to about 60 sites in the US and overseas; universities can apply to Google for funding to run such programs, and they are given a list of suggested workshops topics,


including Unplugged. In 2011 for the first time the University of Canterbury in New Zealand was given a grant to run one locally, so the Unplugged material has come full circle!

6.1 Sponsorship and the New Web Site

Because of the CS4HS connection, the Unplugged project itself managed to get sponsorship from Google at the beginning of 2007. The funding was used to greatly improve the web presence, with its own domain (csunplugged.org, and also redirection from csunplugged.com and computingunplugged.org), new graphics and web design, publicity, more videos, and workshops for educators. Because of the funding there was no longer any need to charge for the books, so the PDF files of the books were made available online for free download under a Creative Commons Attribution-NonCommercial-NoDerivatives license (the NoDerivatives clause was needed because of the existence of commercially published translations.) Initially web hosting was provided by Carnegie Mellon University, as it was better to have a server in the United States rather than New Zealand. More recently this has been moved to a general hosting service.

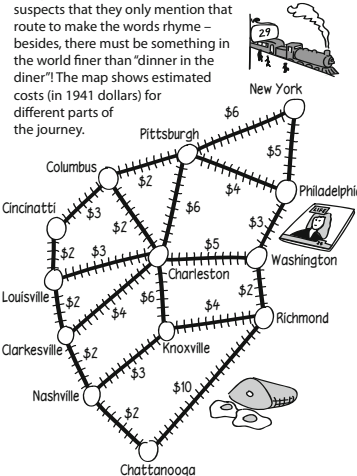
One of the challenges was to publicize the new website; we were aware that having exciting material wasn't enough on its own to motivate a busy teacher or academic to use it. A key target for publicity was the SIGCSE and ITiCSE conferences, where hundreds of passionate computer science educators meet regularly. Figure 7 shows some of the material that was used for publicity at SIGCSE and ITiCSE conferences around 2007–2010. These were giveaways for which the main purpose was to have delegates take away the URL for the new Unplugged site (csunplugged.org). The buttons proved very popular (hundreds were printed and few were left). The “Choo Choo Route Plan” was one of two puzzles that were given to the audience while they were waiting for the keynote session. The main point was to illustrate that Unplugged puzzles could be adapted to a theme—in this case the conference was in Chattanooga, and the puzzle was a shortest-path problem based on elements of the song “Chattanooga Choo Choo.” The postcard (Figure 7(c)) is still used as a give-away at workshops and events so that visitors have something worth keeping that has the URL on the back. The designs for this publicity material were done by Isaac Freeman.

Traffic to the new website (csunplugged.org) has been steady, with peaks now and then as it gets attention on the Internet (for example, in 2008 someone posted Unplugged on reddit.com, and the site had over 50,000 visits within about 24 hours). In 2011 the traffic on the site averaged about 780 page views per day from about 202 unique visitors. About 38% of the visits are from the USA, with about 6% each from India and the UK, and about 4% each from Brazil, Germany, Canada, New Zealand, Italy and Japan. People from 140 different countries accessed the site during the year. The most frequently accessed pages are the activities for binary numbers, image representation, sorting algorithms and searching algorithms, although because the whole book can be downloaded, this doesn't necessarily represent which activities are used the most.

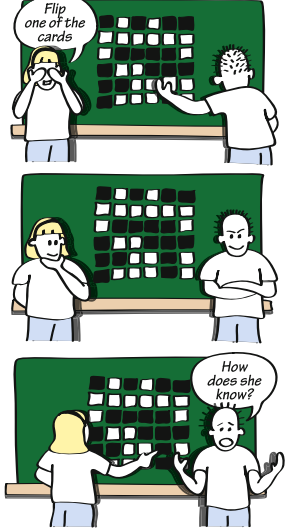


Choo Choo Route Plan

A passenger on a train from New York to Chattanooga wants to work out the cheapest route. He's heard in a song that you should go through Baltimore and Carolina, but he suspects that they only mention that route to make the words rhyme – besides, there must be something in the world finer than "dinner in the diner"! The map shows estimated costs (in 1941 dollars) for different parts of the journey.



Which route to Chattanooga is actually the cheapest?



(a)
(b)
(c)

Fig. 7. Publicity material used for CS Unplugged (a) buttons given away at SIGCSE 2008 (b) the Chattanooga Challenge from SIGCSE 2009 (c) the front of a postcard explaining the parity trick

Experience has shown that the best way to get people switched on to Unplugged is through doing it, not reading about it. For this reason many workshops have been run over the years by the authors, as well as colleagues in many countries, in places as diverse as Bergen, Wuhan, Seoul, Tokyo, Münster, Stockholm, Washington DC, Tacoma, Vancouver, Pittsburgh, Oregon, remote villages in India, Vietnam, Australia, and of course Christchurch, Hamilton and Victoria BC. Mike is especially proud that many in the parameterized community have joined in his enthusiasm by presenting workshops on Unplugged with him in their children’s classrooms, and by finding other innovative ways to open up public understanding and participation in the mathematical sciences. Since the year 2007 workshops on Unplugged have been run annually at SIGCSE, with various people helping to run the workshops and other Unplugged events at SIGCSE including Mike, Frances, Tim, Bengt Aspvall, Peter Henderson, Lynn Lambert, Daniela Marghitu, Ben Tsutomu Wada and Tom Cortina. At a “Birds of a feather” session at SIGCSE in 2008 we were pleasantly surprised to hear how widely Unplugged was being used, including for a one-week visit to an orphanage in Haiti and on trips by float plane to First Nations communities.

We have already mentioned the 2007 SIGCSE workshop which was so popular it had to be repeated. Another memorable workshop was one that was part of the 2008 New Zealand Computer Science Research Students' (NZCSRS) conference in Christchurch. The postgraduate students participated in the workshop at the end of a conference. It began with a demonstration with some local school classes coming in to the university and participating in an Unplugged show. One of the “aha!” moments was when an HCI topic came up—the postgrads were at the back of the lecture theatre, and Tim asked for a show of hands of those who were researching the topic, which greatly impressed the school children (they probably imagined researchers as stuffy old professors rather than young PhD students). The PhD students themselves were impressed that 12-year-old kids could understand the essence of the topic that had been consuming them 24 hours a day for several years! That evening the postgrads had dinner at a science center (Science Alive!, where Tim first started doing computer science outreach), and they were encouraged to “play” on the equipment. Inspired by this, the next morning they got into teams to develop new activities. Several clever ideas came up (including “Harold the Robot”), and at this point we realized the importance of giving people a lot of time in workshops to develop ideas.

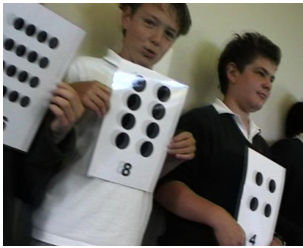
Another memorable workshop was held in Seoul, Korea, run by the “PINY” group, which is a mixture of artistic and scientific thinkers. The workshop began with a visit to a stationery store, where all sorts of papers, card, tubes, balls, wires and other props were purchased. The remainder of the day was spent in the inspiring surrounds of a traditional Korean house, trying to make activities using the materials purchased (Figure 8). Tim would suggest a topic (such as Euler paths), and they would try to make as many activities as possible relating to that from the props available. Photos and information about this event and related ones are on the web at blog.piny.cc/8.



Fig. 8. A PINY CS Unplugged workshop in Seoul, South Korea

6.2 Professional Videos for CS Unplugged

The funding from Google meant that the project team was able to be a lot more creative with video production. The original three videos were done on a zero budget, and although they had proved very popular—for example, the binary number video has had over 17,000 views on YouTube—the production quality was low and a lot was learned about making them suitable for an international YouTube audience. Even with funding from Google, the budgets for the videos were nowhere near what video companies would normally expect for short movies, but fortunately two Christchurch video companies (Shuriken and Orange Studio) were found who were prepared to work on the projects. Between them they produced about three videos per year, mainly using students from Chisnallwood Intermediate School in Christchurch to demonstrate the activities. Some images from the videos are shown in Figure 9. Many of the videos have had translated commentaries or subtitles added, and more recently full high quality versions of the videos have been made available for download from vimeo.com/user6351443. They have also been distributed with the Chinese version of the book. Table 1 gives a full list of the currently available CS Unplugged videos on the Unplugged channel on YouTube (www.youtube.com/csunplugged).



(a)



(b)



(c)



(d)

Fig. 9. Samples from Unplugged videos (a) the first video on binary numbers (b) transmitting an image and decoding it on the side of a school building (c) the “Orange Game” with a variety of fruit (d) “Reaching Out”, with hidden messages coded in binary in the music

Table 1. CS Unplugged videos

Video title	Production team	Comments
Count the dots (Binary numbers)	Tim Bell, Michael Bell, Bengt Aspvall	Commentary in English, Chinese, French, Korean, Swedish
Beat the clock (Sorting networks)	Tim Bell, Michael Bell, Bengt Aspvall	Commentary in English, Chinese, French, Korean, Swedish
Card flip magic (Error detection and correction)	Tim Bell, Michael Bell, Bengt Aspvall	Parity trick, commentary in English, Chinese, French, Korean, Swedish
Computer Science Unplugged — The Show	Orange Studio; Tim Bell, Michael Bell	The one-hour show presented by Matt Powell and Javier Jarquin in 2008, with a commentary by Tim. Polish subtitles available. Also presented on YouTube broken into shorter parts.
Treasure Hunt	Shuriken; Tim Bell, Richard Bell	Finite State Automata, commentary in English, Chinese, French, German, Japanese, Korean, Swedish [26]
Orange Game	Shuriken; Tim Bell, Richard Bell	Routing and deadlock, commentary in English, Chinese, French, German, Japanese, Korean, Swedish [27]
Image Compression (Making Contact)	Shuriken; Tim Bell, Richard Bell	Run length coding activity, commentary in Chinese, French, Swedish [28]
Sorting algorithms	Shuriken; Tim Bell, Richard Bell	Selection sort and quicksort, commentary in Chinese, French, Swedish; Polish subtitles [29]
Computer Science Buskers?	Orange Studio; Tim Bell, Michael Bell, Kristen Finnerty	Parity error correction codes, subtitles in Chinese, French [30]
Santa’s dirty socks (divide and conquer)	Orange Studio; Tim Bell, Michael Bell, Victor Chicha, illustrations by Tim Powell	Story book reading, has an accompanying book that can be downloaded; Polish subtitles [31]
Reaching Out (Binary Codes)	Orange Studio; Tim Bell, Michael Bell	Modem activity with text coded as notes in a song [32]

From 2009 to 2011 the SIGCSE conference accepted videos as submissions, and so the Unplugged videos during that period were submitted and ended up being played at the conference. Because the conference encouraged creativity in the videos, they gradually moved from being a simple commentary on a standard activity to some quite creative takes on communicating the ideas. They include the parity trick being done by a street magician, an animated divide-and-conquer story called “Santa’s dirty socks” (written with Victor Chicha, a visiting intern), the run-length coded image painted on the side of a school building (Figure 9(b)), and an MTV style video where the tune of the song encodes hidden messages in binary using high and low notes (Figure 9(d)).

Some of the videos were tailored to the theme of the SIGCSE conferences; the 2011 conference theme was “Reaching out”, which was also the title of the song that coded the binary messages; and the 2010 theme was “Making contact”, for which the video discussed coding pictures as numbers based on an idea in Carl Sagan’s book “Contact” [33].

Two of the videos (the “Reaching out” song and “Santa’s dirty socks”) have led to two new activities on csunplugged.org to support them. The song is supported by an activity that explains how binary can be transmitted using sound (as on modems), and has a warm-up exercise with a recording of a jazz singer singing short coded messages. The singer had first performed the songs as part of an impromptu exercise at a music education conference; the theme of the music conference was “Modulations,” so a modem exercise seemed appropriate! The “socks” video is about divide and conquer, which now has its own activity, including a picture book of the story that is provided as a PDF file.

Most of the videos are short demonstrations, but the 2008 video of the Unplugged show was intended to help future presenters, with a live recording of the one-hour event, interspersed with a commentary that explained the purpose of the show (and the Unplugged philosophy in general), and hints on presenting it. On 9 January 2011 the YouTube video about the show was picked up on the Reddit recommendation site, and in 24 hours about 20,000 people watched the video, making it the largest audience for an Unplugged event yet! To date the show video has had over 47,000 views, and has drawn a lot of positive comments.

6.3 Translations to Other Languages and Cultures

The English-language Unplugged material has been used internationally since it was first written. The first enquiry about a translation came from Korea University, which had been actively involved in seeking ways to teach “real” computer science in the South Korean school curriculum, as reflected in the title of a 2006 paper they had published called “Informatics Education — The Bridge between Using and Understanding Computers” [34]. The translation was headed by Prof Won Gyu Lee, and the liaison with the Unplugged team was done by Sook Kyoung Choi, one of his postgraduate students who was researching how to adapt such activities to the Korean education system [35].

The Korea University initiative resulted in the first translated version, published in 2006 [23]. Because it was done through a publisher, it also became the first commercially published version of the Unplugged material. Susumu Kanemune, who was a Japanese colleague of Prof Lee, took an interest in the book, and by 2007 had published the Japanese version, with an appendix by Yasushi Kuno providing additional ideas and discussion. Many other translations followed, some prepared for formal publication, but most done by volunteers who were enthusiastic about the material and wanted to make it available to colleagues in their native language; these translations are available through the CS Unplugged web site. Table 2 lists the translations that have been made.

In addition, translations in Bahasa Indonesia, Bengali, Dutch, Hungarian, Maori, Tamil and Welsh have been proposed or started, although because such projects are done by volunteers, often it can take some time until they are completed! A translation into Uzbek was also started at one stage, but unfortunately the NGO doing the translation was asked to leave the country, and the project was cancelled.

Using Unplugged material in other cultures brought both challenges and fresh ideas. Mike had already observed that the “stories” might not make sense in other cultures — for example minimizing the number of ice-cream stands didn’t make sense in Peru if there was high unemployment [37]. When the Unplugged activities were used in Asia some of the examples dependent on language needed to be revised; for example, the 5-bit binary number code for the alphabet doesn’t work so easily for some languages, and some cultural assumptions needed to be adapted [38]. Interestingly, one of the first exercises in the teachers’ edition of the book involves Christmas trees, and Tim checked with each translator whether they felt this symbol from a Christian tradition would be a sensitive example in their country. Most translators (including those from Asia and the Middle East) felt that Christmas trees were generally acceptable and even enjoyed in their country, and intriguingly the only pushback came from a country with an English heritage... the USA!

As mentioned earlier, the title in translations usually keeps the word “Unplugged” in English to avoid pejorative meanings in the local language. Choosing a suitable subtitle has needed local input. Usually it contains the word “games” as this would be considered an attractive feature, but in Asian countries, we were advised to avoid the word because parents concerned about their children getting a serious education might shun a book that purported to be fun! Of course, the books still contain games, and fun is an important part of learning, so it’s almost as if the fun had to be smuggled in to the students between covers that parents would approve of.

Most cultural problems were easily overcome once recognized, and in general translations and internationalization also brought cultural richness and fresh ways to present the activities [38]. For example, Chinese colleagues suggested strings of lanterns for binary numbers, and Japanese colleagues introduced us to double-sided magnetic sheets for doing the parity trick on a whiteboard.

An example of a culturally adapted Unplugged activity is shown in Figure 10, where the binary number activity involving Christmas trees has been used for a “Fujitsu Kids Event”; as well as translating the instructions, the Christmas trees are now lanterns on a string, the alphabet has been changed to hiragana, and the cartoon characters are Manga style, which is appealing for Japanese students. The code table used is limited to 32 characters, which are sufficient for the particular message being coded, but a 6-bit code would be needed to code all hiragana characters to allow any message to be represented.

The Chinese edition, published by HUST press in 2010 [39], was a heavily re-written version aimed at students rather than teachers.

Table 2. Translations of CS Unplugged

Language	Translator(s)	Status
Arabic	Mohammed Obaid	All activities translated, seeking publisher, two activities available on csunplugged.org
Chinese (Simplified)	Muzhou Xiong, Zhensong Liao, Su Yu, Wang Shenglan, Han Ying Chun, Xie Xia, Dong Rongsheng	Student edition with 15 topics available for purchase from HUST press www.hustp.com, original edition translated, most videos translated, csunplugged.org site translated
Chinese (Traditional)	Long-Yuan Ya	Several activities translated
French	Francois Rechenmann, Paul Gibson, Anne Berry, Isabelle Souveton, Victor Chicha	Teachers' edition available with preface by Roberto di Cosmo through Interstices site interstices.info, and csunplugged.org, most videos translated
German	Maexl Stege, Katrina Kranzdorf	Five activities available on csunplugged.org, two activities used for www.informatikjahr.de
Greek	Constantine Mousafiris, Theophanis Hatz	Teachers' edition available through activities at csunplugged.org
Hebrew	Benny Chor, Simon Schocken	Being published as a blog at csu-il.blogspot.com
Italian	Giovanni Michele Bianco, Renzo Davoli	Teachers' edition available on csunplugged.org
Japanese	Susumu Kanemune, Yasushi Kuno	Teachers' edition available for purchase from Etext publishers, www.etext.jp/unplugged.html [36]
Korean	Won Gyu Lee, Sook Kyoung Choi, Hyeoncheol Kim	Teachers' edition available for purchase from www.yes24.com [23]; most videos translated
Polish	Pawel Perekietka, Lukasz Nitschke	Teachers' edition available through activities at csunplugged.org, subtitles available for Unplugged show and some other videos
Portuguese (Brazil)	Luciano Porto Barreto	Teachers' edition available through activities at csunplugged.org
Russian	Irina Derevianko	Teachers' edition available on csunplugged.org
Spanish	Alfonso Rodríguez, Lorena Mendoza, Clara Eugenia Garza	Teachers' edition available from csunplugged.org
Swedish	Stefan Hellberg, Bengt Aspvall	Teachers' edition available from www.ide.bth.se/~bia/unplugged/UnpluggedTeachersSwedish.doc, web site at www.bth.se/csunplugged, most videos translated
Turkish	Sertan Girgin	Seven activities available through csunplugged.org

6.4 Adaptations and Variations of Activities

In the early days of Unplugged we had expected that the number of activities would continue to grow. Although new ideas did come up, the real growth has been in variations of activities and taking them into different contexts.

For example, Figure 11 shows some of the examples that followed from the binary number card activity, which was the first activity in the two main Unplugged books. Over the years several follow-up strands of thought developed. The first is a discussion of how birthday cakes actually use base one (unary), and that a binary system for candles would be more efficient and safer (Figure 11(a) and (b)). This can lead on to discussions relating to the logarithmic/exponential relationships that permeate computer science and both excite and frustrate algorithm designers. Dividing the cake itself can illustrate the power of binary divide and conquer approaches: give half the cake to the first person, half of the remainder to the next, and so on. People down the line soon realize how efficient logarithmic complexity is, even though the word “logarithm” isn’t mentioned! Another practical application of binary numbers for children is coding names and words into jewelry using binary (Figure 11(c)).

A variation of the orange game that was created in two places independently was the idea of using different colored fruit instead of labeling the oranges, and having students wearing t-shirts corresponding the fruit. This was proposed by both Richard Bell (making a video of the activity) and Gottfried Vossen (running a “children’s university” in Germany).

These kinds of variations have gradually arisen as the Unplugged material was used over the years and others contribute ideas.

Another significant change has been the adaptation of Unplugged for different contexts. In some places it has been used with little adaptation, such as having activities catalogued in the CSTA repository of teaching materials. Other times some local expertise has been added to improve the suitability of the activities. For example, the CS4FN (Computer Science for Fun, cs4fn.org) has a book of magic tricks, and the parity trick appears in their book in a more flamboyant version: the demonstrator is blindfolded right from the start, so they never get to see the original cards that the volunteer puts down, making the trick even more impressive. The National Center for Women and Information Technology (NCWIT) promotes “promising practices” to people wishing to communicate ideas from IT to attract and retain female students, and one of their promising practice handouts features the sorting network activity, and promotes Unplugged in general. They also have a “programs-in-a-box” series, and 7 of the Unplugged activities have been adapted and turned into NCWIT’s “Computer Science in a box: Unplug your curriculum” (www.ncwit.org/unplugged). These activities are very similar to the teachers’ edition, but are mapped better to the US curriculum. In 2006 some of the Unplugged activities were translated into German and packaged with other material for their “Informatikjar” (Computer Science Year). Andrea Arpaci-Dusseau has been working on a parent-child version of

① コンピュータのことで遊ぼう！ (2進数)

復習問題 チャレンジ 1

富士通キッズイベント2008
Fujitsu Kids Event 2008
夢がかなう時間らしく

秘密のメッセージを送ろう 1

ユウタはデパートの屋上に閉じこめられてしまった。見たいテレビもあるので早く家に帰りたい。どうしたらいいだろう？ 声を出してさげんでみたが、だれも近くにいないようだ。道の向こう側では、夜おそくまでコンピュータの仕事をしている人たちが見えた。彼らに気がついてもらう方法はあるだろうか。ユウタは使えそうなものがないかさがしてみた。そしてアイデアが浮かんだ。祭りちょうちんの明かりでメッセージを送ろう！ 彼はちょうちんの明かりをつけたり消したりできるようにコンセントを探した。そして道の向こうの女性が理解してくれそうな、簡単な2進数のコード (符号) を使った。うまく伝わるだろうか？



- ① 祭
- ② 祭 祭 祭
- ③ 祭 祭
- ④ 祭 祭 祭
- ⑤
- ⑥ 祭 祭 祭
- ⑦ 祭 祭 祭
- ⑧ 祭 祭 祭
- ⑨ 祭 祭 祭
- ⑩
- ⑪ 祭 祭
- ⑫ 祭 祭
- ⑬ 祭
- ⑭ 祭 祭 祭
- ⑮ 祭 祭 祭 祭 祭

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
あ	い	う	え	お	か	き	く	け	こ	さ	し	す	せ	そ	た
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
ち	つ	て	と	な	に	ぬ	ね	の	は	ひ	ふ	へ	ほん	ん	っ

こたえ

①	②	③	④	⑤	⑥	⑦	⑧	⑨	⑩	⑪	⑫	⑬	⑭	⑮

Fig. 10. A page from activities for the Fujitsu Kids Event, 2008

Unplugged, intended as a workbook for a non-specialist parent to work through with their child. This requires more hints for the parent, and also requires some of the group games to provide versions that can be played by two people (for example, turning the sorting network into a board game).

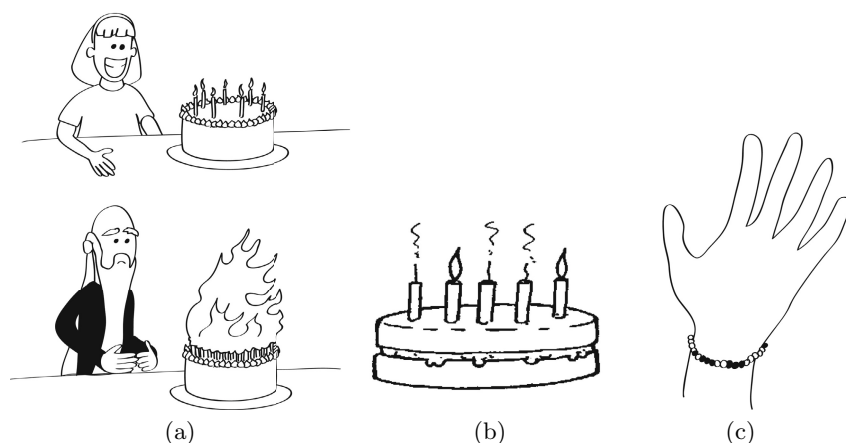


Fig. 11. Binary numbers in everyday life for kids: (a) the dangers of using unary (base 1) instead of binary (b) the binary birthday cake (c) a bracelet coding a name using beads

Unplugged activities have found their way into formal curricula. The “Computing Science Inside” program hosted at the University of Glasgow provides activities that are ready for classroom teachers to use, and is particularly aimed at the Scottish curriculum; it contains several Unplugged activities that have been repurposed for this use. The resources include Powerpoint slides and hand-outs (see csi.dcs.gla.ac.uk). The “Exploring Computer Science” (ECS) program in Los Angeles (www.exploringcs.org) is a successful initiative that uses several Unplugged activities as part of its curriculum [40,41]. Another initiative that is likely to draw heavily on Unplugged activities is Peter Denning’s “Computer Science Field Guide” (www.csfieldguide.org), which is planned to be a merit badge system where participants can earn “badges” for achieving competency in a selection of areas, and at different levels [42].

Around 2006 the idea of Computational Thinking (CT) became prominent in discussions about computer science education, championed by Jeanette Wing [43]. It turns out that CS Unplugged is a good example of an approach that emphasizes CT, and was already in widespread use when CT became a hot topic. Unplugged has been explored in a 2010 workshop on Computational Thinking [44].

A more significant adaptation that is now underway is to make the material more suitable for direct use in the classroom. The teachers’ edition was written as extension exercises, but for everyday use teachers need background reading for the students (currently the books are written for the teacher, not the students), and assessment material (the nice thing about outreach is that you don’t usually have a test at the end!) A version intended for students has been written for the Chinese market [39], but is yet to be adapted for English speaking countries. This version has a teacher guide, and the main text is addressed to the student, being careful to cover basic information that would be expected in the classroom (such as defining kilobytes and megabytes) as well as the more open-ended material

like the muddy city puzzle. As the Unplugged material gets wider adoption and is used in the formal school setting, it is becoming apparent that the original open-endedness and sense of adventure will take some creativity to retain, as schools around the world feel an obligation to standardize and assess, and there is no guarantee that the teacher will have the passion or experience of math that someone running an outreach program would.

Although the Unplugged philosophy generally eschews using digital devices, there has been value gained by integrating it in some situations. For example, Daniela Marghitu has had students design and program robots to carry out Unplugged activities, so they need to understand the activity first, and then program the robot to simulate the actions of a child who would have been doing the activity, which requires an even higher order understanding of the concept. These activities have been done as part of “Robo camp”, a robotics program for advanced students ages ten to eighteen [45]. Moti Ben-Ari has also demonstrated how Unplugged activities can be followed up with Scratch programming exercises by implementing many of the activities in Scratch (available from code.google.com/p/scratch-unplugged/), and Ward *et al.* also give hints for using Unplugged with Scratch [46].

In 2008 the “New Media Consortium” (NMC) awarded a prize of US\$5000 to the Unplugged project to develop activities in the Second Life virtual world. A sorting network was implemented (it can be seen in Figure 2(e)) and was used for a period by students, including some with disabilities who couldn’t walk in the real world. Although this might appear to be cheaper and simpler than a physical sorting network, the ongoing cost of virtual land to put the sorting network on, and a lack of ongoing funding and support, prevented it becoming a public facility. Work is still continuing on the possibility of virtual worlds for implementing and evaluating Unplugged activities, although it has moved to private areas that can be hosted within a school, which avoids the many issues that surround school children using a public virtual space [47].

Yet another adaptation of the material is for a programming competition environment [48]. By using Unplugged stories as scenarios to be solved in a competition the programmers end up having to grapple with deep issues from computer science, but also there is the potential to use the physical activities as a break from working intensely at the keyboard, where the break itself provides the next programming challenge!

At the University of Canterbury some of the Unplugged activities have been landscaped into a “Bridges of Friendship Math/Computer Science garden”, which includes the seven bridges problem (Eulerian path), an 8-queens puzzle, and of course, a 6-way sorting network (shown in Figure 2(c)). Running around the seven bridges trying to find an Eulerian path provides a valuable break from lecture theatre activities for school visits, and even regular student classes.

A new development with Unplugged has resulted from abstracting the principles of this approach to teaching computer science, and thinking about how it could be applied to other subjects. If the meaning of “Unplugged” is to remove traditional gateways to the subject, and enable young children to grapple with

advanced ideas before learning the “basics,” what would the equivalent be in a different subject, such as music? The first task is to identify the gateways — perhaps having to learn a musical instrument, or read music notation, before one is allowed to become a performer or composer? Such ideas are being explored in workshops, including the transdisciplinary environment of the SDPS conference [2], and a workshop under development called “It is Really About Thinking.” This kind of approach can be found in the book “Statistics Without Math” [49], which provides a gentle explanation of statistics based on diagrams, which in turn might provide motivation for a student to tackle the mathematics behind the concepts. It will be valuable to explore these principles in other disciplines; who knows how many educational opportunities could be opened up this way!

7 CS Unplugged—Emerging Principles

With 20 years’ experience, much has been learned about delivering computer science using the Unplugged approach, although Mike’s original writings about computer science and math education still emerge as the key principles that have stood the test of time.

There were several motivators for Unplugged, which was initially mainly done as a labour of love. The philosophy that drove the project was largely captured in Mike’s paper “Computer SCIENCE and Mathematics in the Elementary Schools” [4], which made the radical proposition that elementary school students could enjoy learning about algorithms, and that computer science was a “treasury of accessible, colorful and active mathematics”. The paper points out that grade 1–4 students mainly get to do arithmetic, and arithmetic is not mathematics. It then demonstrates how such students could easily work with concepts from graph theory, sorting networks (of course), and knot theory. The paper discusses why such inspiring math might not be welcomed in the education system, and also demonstrates some examples of young children contributing to or inspiring serious avenues of research in math and computer science, including an example of a paper that was published as a result of a classroom visit. There are many inspiring quotes and analogies that challenge traditional thinking about math and computers in schools, and it is just as relevant now as it was 20 years ago. It should be compulsory reading for everyone involved in computer science and math education.

Another motivation of Unplugged and related projects has always been to make the ideas from computer science available to those who can’t afford a computer, or might not even have a reliable power supply to run one. Mike was active in working with teachers and students in such situations, particularly through the Kovalevskaja Fund, which Neal Koblitz discusses elsewhere in this book.

Both motivations speak of wanting children to be empowered to understand and reason about the world they find themselves in, and not be mere users of technologies that are imposed on them or are inaccessible to them. It has been

said that “Only two industries refer to their customers as ‘users’: computer design and drug dealing.⁶” The *MEGA-Math* and Unplugged initiatives can largely be seen as a desire to rescue children from becoming only users, that is, becoming addicted to whatever technologies are inflicted on them, rather than being given the wherewithal to create systems that work for them; to choose between what is good and what is harmful; and to discern what will improve their quality of life, and what will improve someone else’s quality of life at their expense.

Neal Koblitz shares these sentiments in his article “The case against computers” [50], in which he quotes Mike as saying “Most schools would probably be better off if they threw their computers into the dumpster” (a phrase which many people would have heard Mike say!) Neal also mentions Mike’s use of the term “Cargo Cult” to refer to the “fetishization of computers by the media and educational establishment.”

Mike wrote and spoke frequently about these issues. It was worded particularly eloquently in an article with Ian Parberry in 1993: “SIGACT trying to get children excited about CS” [51], which said:

We need to do away with the myth that computer science is about computers. Computer science is no more about computers than astronomy is about telescopes, biology is about microscopes or chemistry is about beakers and test tubes. Science is not about tools, it is about how we use them and what we find out when we do.

The analogy between computers and telescopes is a particularly powerful one that has served well to make the point quickly to laypeople about the difference between computer *science* and simply learning to use computers. As an interesting side note, it appears that the telescope analogy originated with Mike (for example, he used it in a 1991 online pre-publication of his “manifesto” document [4]), but searching the Internet shows people widely attributing it to Dijkstra, without any reference to where Dijkstra first said or wrote it (and it doesn’t appear in his collected writings). Mike and Dijkstra had spent time together around the time that Mike was writing this material, so it is quite likely that it was discussed and possibly even originated from those conversations. It would be an interesting research project to settle the history of what has become such a definitive quote. Some initial research is reported via en.wikiquote.org/wiki/Edsger_W._Dijkstra, which concludes that the quote is misattributed to Dijkstra, and is actually from Mike. Establishing this without doubt would provide an extreme example of the spread of misinformation by copy-and-paste reporting on the Internet!

Another quote from Mike that left an impression on Tim during the 1993 visit was “Computer science is the rock and roll of mathematics.” This inspires all sorts of imagery — is computer science the part of mathematics that people actually use every day? Is it the part that causes things to happen? Is a computer scientist looked down on by a “pure” mathematician because they make compromises to make things work for everyday people? Like so many of Mike’s

⁶ This is usually attributed to Edward R. Tufte.

epigrams, it's a concise statement that gives people pause for thought, and enables someone speaking to laypeople to communicate a lot of meaning in a very short time.

As Unplugged became popular, the following features emerged that defined its value as an approach to education and outreach:

- Teaching a student to program takes many hours, if not months. If you only have one hour to spend with students (e.g. an outreach visit), then the Unplugged approach enables the presenter to launch into a range of computer science topics, rather than just scratch the surface of programming. Extending Mike's analogy, if an astronomer had one hour to spend with a class, it wouldn't be inspiring if it was simply some preliminary information about how to set up a telescope.
- Programming is usually put up as a gateway to getting into computer science. Some students may not particularly enjoy programming, but would be prepared to use it if they knew what they could do with it. The way computer science is often taught, students get the impression that it is primarily about programming; Unplugged reverses this view.
- It is often assumed that the first thing you need to do computer science is a computer, but often a large amount of effort can be spent setting up a computer lab, installing appropriate software that will soon need updating, creating accounts for students, and so on, making the computer a distraction that preoccupies both students and administrators. And of course, in some countries there might only be a few computers in the whole school, or maybe none at all. Unplugged activities enable learning about the discipline to occur in the first minute the lesson starts, and only inexpensive equipment is needed — typically paper and pencil, and maybe some string!
- There are many misconceptions about what computer science is, and the way it is often approached in schools can reinforce them. We encounter many students who had assumed that computer science could not possibly be an interesting career for them, yet somehow got into it by accident, and find it thoroughly fulfilling. It would be a tragedy if students decided not to follow a path that they would have loved based on misconceptions; Unplugged provides a means to sweep away many of these preconceptions.
- Information systems have a huge influence on everyday life, whether or not people are interested in computers. People who don't understand even simple computer science concepts have to make important decisions relating to the security of their computers, the way they do financial transactions, or whether a technology is reliable, based mainly on the opinions of others. It would be equivalent to supporting the invasion of a foreign country based on friends' opinions, rather than understanding the culture and politics of that country and making an informed decision.

A key theme in *MEGA-Math* and Unplugged is the sense of story—there are pirates, monsters, ice cream vendors and football teams who are at the centre of some story that invokes a problem that must be solved.

In Mike’s “advice to students” at mrfellows.net he writes:

Story is central. Story is a bigger force than science. Everybody lives by stories. They are a primal force. In mathematics, we add formalism. We have equations that lead to solutions but story has its own logic. Find the story in what you are telling and presenting. This will help the listener meet you more than half-way.

Some might think that fictional, even preposterous, stories have no place in teaching science and mathematics. But stories engage children and adults, they provide a compelling description of a situation, they remove boundaries, and they give the message that it’s time to start using your imagination. Mike is a consummate story teller, and has been able to use story to great effect, as well as providing stories that others can use, if only they are prepared to suspend reality for the sake of science.

A more prosaic analysis of the Unplugged approach can be found in a paper initiated by some Japanese Unplugged enthusiasts, who analysed Unplugged activities and came up with a *design pattern* for the activities [52]. The paper explores mapping everyday objects (such as cups and stickers) to concepts in computer science (such as variables and states), and gives ideas for creating new activities.

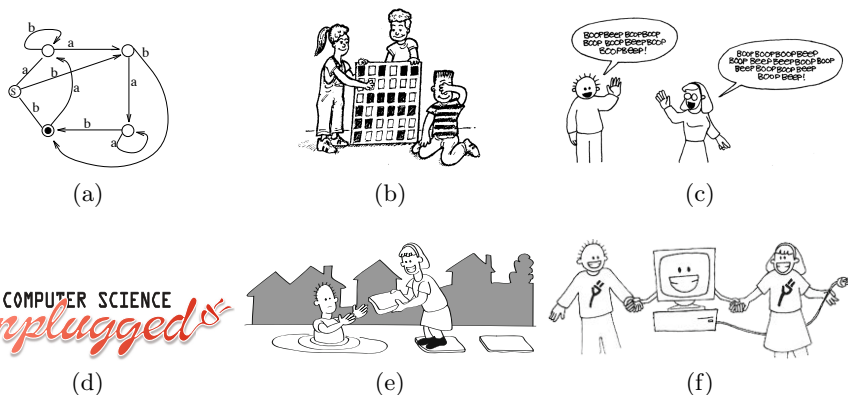


Fig. 12. Illustrations for CS Unplugged (a) early material from *MEGA-Math* (b) from the “original” 1999 book, by Gail Williams (c) from the teachers’ edition, by Matt Powell (d) the logo designed by Matt Powell (e) from the web site by Isaac Freeman (f) theme image from recent books and web site

Having good illustrations has been an important aspect of the Unplugged project. The *MEGA-Math* workbook [8] used line drawings of diagrams without any people shown in the images (Figure 12(a)), although the *MEGA-Math* website includes low-resolution images including some characters. Early illustrations for CS Unplugged were done by various computer scientists involved

in the project, with some help from Malcolm Robinson (a Christchurch-based graphic artist), but for the book published in 1999 most of the illustrations were done by Gail Williams, who introduced illustrations of children doing the activities (Figure 12(b)). The teachers' edition was illustrated by Matt Powell, a CS graduate who captured ideas from CS in the illustrations, and also created the characters that have become the familiar face of Unplugged (Figure 12(c)). He also designed the logo (originally for the KidsFest show in 1998), which is still used (Figure 12(d)). More recently the illustrations have been taken over by Isaac Freeman, who has continued developing new illustrations in the style established by Matt Powell (Figure 12(e)).

8 Evaluations of CS Unplugged

The ideas in CS Unplugged have clearly had a wide impact around the world, and a search of the computer science education literature reveals that it is cited in dozens of papers. In October 2011 the website was getting nearly 12,000 unique visitors each week, and the various videos on the YouTube channel have had over 80,000 views in the few years they have been available. The widespread adoption of the ideas in many countries is an endorsement of the material, but formal evaluations are important to understand more carefully how well the approach works, and in what contexts it doesn't work. Although many evaluations of Unplugged exist, they either use very small groups, or mix Unplugged with other material, which means that it is difficult to draw general conclusions from them. More often than not, teachers have quickly recognized the effectiveness and value of Unplugged and adopted it, proving its worth in practice for themselves. However, there are many factors that can affect how well it works, including the time of day (children seem to have a better attention span in the morning), the enthusiasm of the presenter and especially their patience to let students explore ideas.

There are two research projects that report on using Unplugged in its raw form with older school students. Taub *et al.* [53] report on a group of thirteen 7th and 8th grade students who did 18 of the 20 activities from the "original" book as a series of after-school meetings. Six of them were then interviewed; the sample is too small to draw firm conclusions, and it's unfortunate that the original edition was used because some of the problems encountered were addressed in the revised teachers' edition, but the paper does make the following useful observations:

- Activities should build on students' prior knowledge (Unplugged was designed for elementary age children who would have a much less sophisticated prior experience of math and computing, and the teacher should adapt activities to suit the students' background),
- There should be an explicit link to central concepts in computer science (some of the activities were intended to be exploratory, to follow the students' interest, but if Unplugged is to be used as a high school text book then more formal links to curriculum would be needed), and
- students should be informed about careers in computer science (this was seen as an important motivator, although is clearly complementary to the material presented in Unplugged).

Another study by Feaster *et al.* [54] describes an outreach to a high school where they ran 10 lessons from Unplugged activities for two groups of 14 and 15 students respectively. The paper reports that those who started with low interest increased their level of interest, but those with a high level didn't; that is, Unplugged seemed more suitable for getting students interested than for those who were already interested. After the sessions the students seemed to have a better idea of what university CS would be (for example, they decreased their belief that web design would be important preparation for studying CS, and saw math as more important for CS). The authors observed that high school students didn't seem as excited about these kinds of activities as more junior students; this reflects our experience, where for senior classes Unplugged is best used as a short demonstration followed by a more technical discussion, whereas younger children are content to spend more time exploring ideas for the sake of it.

A more positive picture emerges when the material is used to add interest to camps and outreach programs. Carmichael [55] reports using Unplugged interspersed with teaching video game programming for a summer camp, and achieving an overall increase in interest in taking CS further as a subject. In this case the Unplugged activities apparently helped students see the connection between theory and practice.

Hart *et al.* used Unplugged to link CS to the math curriculum in a 3-day workshop for math teachers [56]. A survey showed that 100% of teachers agreed or strongly agreed with the statements "Sessions stimulated my interest", "Content is useful to me", and "Program will improve some aspect of my teaching." One teacher commented: "I loved getting a little 'taste' of many different aspects of the CS field. I now have some first-hand information about the CS field to pepper my lessons with throughout the year."

A computer science outreach program for fourth graders based on Unplugged activities reported success in increasing interest in computer science [57]. They noted that it is hard to get public schools schools interested in such events because of the lack of computer science state standard tests. From pre- and post-intervention surveys they reported that "students were more interested in computer science, had significantly higher cognitive competence, and were significantly more confident about math ($p < 0.05$ for all), but not significantly more interested in math."

Groover reports success from activities with girls in a middle schools "conference" which included Unplugged activities [58]; the conclusion says that "post activity discussions showed that the students seemed to understand the connection of each activity to computer science." Interestingly, the "Marching Orders" activity was more popular than the Parity Magic trick; our experience is that the parity trick usually generates a lot of interest, which illustrates how experience can vary a lot in different contexts.

The Parity trick has also been used as part of a successful program based on magic tricks [59]. Feedback from this magic-based program indicated that a significant number of students indicated that they learned something about

computer science. Having the presentation as a “show” rather than a “lecture” was seen as positive, and the authors report that it “clearly worked for girls.”

One of the main points of Unplugged is to change attitudes, and a research project that measured this was reported by Cottam, Foley and Menzel in 2010 [60]. They organised a roadshow that talked to students about topics like stereotypes and careers in computing, and the main computing activity was the Parity trick. The evaluation was for 613 freshman students (59% female), and the results were generally positive. The question asked of students that relates most to understanding the field was “Computing is mostly about writing programs”; 24% of students agreed before the intervention, and 17% agreed afterwards. In response to the statement “Computing is full of exciting opportunities”, 55% of students agreed before the intervention, and 81% agreed afterwards. A larger change was seen for female students: 49% of females agreed before the intervention, and 79% agreed afterwards.

Another report on a workshop for teachers that uses Unplugged as a major component found that the attendees “felt much more comfortable advancing the use of computing and computational thinking in their classes” after the workshop [61]. In a survey 4 to 6 months after the workshop, one of the three most widely adopted materials from the workshop were the Unplugged activities (particularly the binary numbers activity).

A program that combines parents and students in a series of workshops is reported by Hart [62]. The target audience is fourth through sixth grade female students. In the workshops they covered a variety of topics including programming in Alice. Unplugged was used for an introductory session, during lunches, and with the students while parents had sessions on topics of less interest to the students. It is difficult to draw conclusions about Unplugged from this report because it was an adjunct activity to the main program.

A comprehensive study of the Unplugged approach can be found in a thesis by Sarah Carruthers [63]. The thesis reports on a series of lessons with sixth grade students relating to graph theory. A key conclusion is that the students “appear capable of not only learning graph theory, but applying it to solve problems. The use of relational graphs appears to positively impact student performance on at least some types of problem solving activity.” In a related paper, Carruthers *et al.* [64] conclude that “Graph theory instruction can support existing mathematics curriculum and provide novel problem solving strategies for students at the grade six level. Student and teacher willingness to participate actively during the graph theory lessons in this study indicates that graph theory may be a suitable computer science topic to integrate in classrooms at this level.” This matches with the anecdotal experiences reported early in this chapter—as long as you don’t tell students that it is difficult, they are able to work with concepts that are generally regarded as very advanced.

A survey of Unplugged and several other approaches to outreach and teaching that avoid programming was published in 2011 [65]. One issue that was identified was the importance of a suitable motivation for students. For classroom work

the extrinsic motivation of grades is usually available, but for other programs motivators include “contest prizes, the challenge of solving a problem, curiosity, humour, and ideally, appealing to the intrinsic interest of the student in this kind of thinking and reasoning.” The challenge to any presenter, whether using Unplugged or something else, is to create those motivators so that children want to explore the concepts, rather than complete the work out of necessity.

Based on the evaluations mentioned above, Unplugged has a positive role to play for adding seasoning to classes and outreach programs, and has been well received in workshops for teachers. It works well in combination with other topics, including programming or exploring social issues or careers. Problems have been reported using it directly as a curriculum, which is not surprising since it was developed in the context of outreach and open-ended exploration, rather than a specified set of standards. Its genesis was from an inversion of the normal classroom format of “teach an algorithm and test if the students understand it” approach, and the research reported above seems to confirm that the way in which material is approached in the classroom is as important as what the material is. In terms of the astronomy metaphor, simply taking the Unplugged activities into a classroom and using them to teach a whole course on computer science would be analogous to teaching a course on astronomy without using a telescope at all. An inspired teacher could pull it off, but in a conventional setting, students are bound to want to look at the night sky for themselves, and for a teacher to insist on not using a telescope could be demotivating, particularly for those who want to be an astronomer one day!

There are two main challenges with introducing Unplugged to a formal curriculum:

- poorly prepared teachers can make the most exciting material become dull because the topics are taught under compulsion, and students who come up with interesting ideas might be squashed either because it doesn’t match the curriculum, or because the teacher doesn’t have a broad enough background to recognize creative answers. For example, Tim once observed some science demonstrators doing a binary number activity, and an enthusiastic child hypothesized that there might be a card with 256 dots for the 9th bit; the demonstrators essentially told the child that you can only have 8 bits. On another occasion while doing the first step of selection sort with a class using a balance scale, the students had just found the heaviest of 10 weights, and were asked them how many comparisons were made. Normally students work out that it is 9, sometimes incorrectly suggesting 10 as a possibility. On this occasion, one girl said “it’s going to be 45”. One worries that if they had a teacher not familiar with the problem then their answer would simply be dismissed as being way out of range, instead of recognizing that the student had thought ahead and calculated the number of comparisons for a complete selection sort.

- the school system (administrators, teachers, students and parents!) expect teaching to be followed by assessment. The organic and exploratory nature of Unplugged makes this difficult because one doesn't know what the students are going to learn, and if teachers are required to straight-jacket the material into a strict curriculum, it can squeeze the life out of it. This is not to say that assessment isn't possible, but inspired teaching and creative assessment is required to keep students excited.

9 The Heart of Puzzling: Mathematics and Computer Games

Mike wanted to bring an appreciation for mathematical foundations and frontiers to as many people as possible, and computer games were part of his vision early on. Mike ran his *MEGA-math* theory awareness activities in his classrooms and at Family Night events, and talked about them at the professional meetings he attended, wherever he went. Computer games were galvanizing attention. In 1992, when Mike chaired Local Arrangements for STOC in Victoria and organized an education action committee (see Section 3 on Activism), he met Ernie Brickell, now Chief Security Architect for Intel Corporation. This led to an invitation to speak at Sandia National Labs. Afterwards, Mike visited Vance Faber, Chief Scientist and Director of Research at Los Alamos National Labs. Vance asked Mike what he had been doing in Albuquerque. Mike showed him the manuscript that he and Nancy Casey were writing about teaching mathematics with game style activities, and Vance shouted, "That's great! I've got to show Bonnie Yantis. We've got to turn this into a project." "Research On Mega-Math: Discrete Mathematics And Computer Science for Children" became Vance's favorite project. One of the nine objectives of the project was "The development of connections to the computer games industry."

9.1 A Systematic Mathematical Theory of Game Design

In 1992, Mike offered a course at the University of Victoria (BC) called "Theoretical Computer Science With Applications to Computer Games." By 1993, he had developed a premise that the heart of every good game held both a puzzle and an interactive structure, both inherently mathematical in nature. He proposed that if properly exploited, the addictive fascination of computer game-play would reveal itself to have a fundamental similarity to the mental experiences of mathematical research. The joy of an activity akin to research (carefully constructed gaming) could be a catalyst to popularize mathematics for children.

Mike heard about the Games Development Conference, the largest and primary forum for those involved in the development of interactive games. Insiders gather here to exchange ideas and shape the future of the industry. Mike phoned the organizers and described some of his ideas about math games for children. They said that was very interesting, and they organized Mike to speak in 1994.

This led to consulting with the big game company Broderbund (producer of *Myst*), and Mike was recruited to a contract with them. That was before they closed their education division.

In his unpublished 1996 article, “Fifteen MEGA-Math Puzzles,” Mike described a systematic mathematical theory of game design, and a method of producing an almost endless stream of computer games. His method merged what he termed *vanilla* puzzles (an all-purpose, general ingredient) with game *handles* (that is, a crank, as on an old-fashioned coffee grinder.) The vanilla puzzles came from computational problems such as GRAPH COLORING, DOMINATING SET, INDEPENDENT SET, HAMILTON CIRCUIT, and others from the Garey and Johnson compendium [1]. Each of these vanilla puzzles would form the mathematical backbone of a successful game. Importantly, GRAPH COLORING and all the other computational problems come from significant and interesting applications in the real world. The game’s storyline could be built around these relevant applications. The storyline could be another attraction for players to form an interest in mathematical science.

The *game handles* came from the underlying information-action puzzle structure that Mike identified in many popular games. He grouped popular, well-known games into *families* according to their structure, that is, their handles. As an example, there was a *Family of Discrete Repairs Puzzles* (such as “Minesweeper”), and a *Family of On-Line Puzzles* (such as “Tetris”). Mike’s insight was that in order to design a collection of computer games (as he described it): “one had only to attach the *game handle* to a *vanilla problem* from the NP-catalogue, and turn the crank.”

As an illustrating example, consider the popular game of Tetris. From a mathematical point of view, Mike considered the Tetris-handle as an on-line two-dimensional packing problem. Attaching the Tetris-handle to the vanilla problem DOMINATING SET or the vanilla problem GRAPH COLORING would produce the game “Tetris of Dominating Set” or “Tetris of Graph Coloring”. The “on-line” meant that as the graph scrolled downward, a player would use a limited but replenished supply of tokens to indicate a dominating set or a proper graph coloring. If an un-dominated vertex, or improperly colored vertex, hit the bottom of the screen, the player loses.

The game story might vary from the unadorned scrolling graph of dots and lines, to a story about the practical importance or history of the underlying computational problem (the vanilla puzzle), or it might simply be a fanciful story where the vertices represent goldfish, which must be properly colored before the water in their tank drains away. Mike’s vision of computational puzzling and educational game design is summarized in his 1999 article, “The Heart of Puzzling: Mathematics and Computer Games” [66].

9.2 Designing Games with Jim Andrews

In 1995, Mike met Canadian poet-programmer Jim Andrews at the Mocambo Coffee Shop in Victoria, British Columbia, where Mike was trying his hand at stand-up comedy. Mike says:

I have ambient literary impulses from time to time. About when I was moving to UVic, I was having a bit of an existential crisis and decided not to do computer science anymore. I was going to do poetry and stand-up comedy. I didn't get very far.

Mike and Jim (who have become a lifelong friends) began designing a game that was part educational and part entertainment, generally termed “edutainment.” Their game, “Wordstones,” was based on genetics, in which the player creates creatures, machines, and activities (using finite automata) that mimic the behavior of how ribosomes process strands of DNA.

Although Wordstones takes place in an imaginative realm, one of the rewards of the game is that the biological science is eventually revealed to the player. Mike and Jim also designed CoLoRaTiOn, about graph coloring (see vispo.com/software/coloration/CoLoRaTiOn1.0.exe; Jim also designed vispo.com/arteroids).

Jim recalls those times as follows:

We would get together at my apartment, mainly. This was in 1995 to 1997. If we sat down for a two or three hour session, Mike could come up with at least a half-dozen new ideas. It was a matter, for him, of simply turning his attention to any part of the mathematics of computer science and contemplating the challenges of research, the goals and, often, real-world situations from which the algorithms and problems/theory arose. It was a mark of his work to invariably involve some sort of unsolved problem in the games/puzzles. This is very Mike. It wasn't sufficient to simply create fun puzzles and games. They should also encourage research into productively mysterious, unsolved mathematical problems and issues. The unknown, for Mike, is where the real fun is.

It all needed more work, to be frank. But what fun it was. For me and, I hope, for Mike. I was continually amazed by his dynamic insight and powerful ability to discover games and puzzles in the math of computer science where others saw none. It was a Pythagorean experience for me, as it were. A bit *mathematikoi*, a little *accusmatici*, in awe as I was, of his mathematical insight and sweeping vision of the field.

I also saw Mike's genius at work in his creation of materials for kids to explore and play on during evenings where he and his volunteer graduate students and also other math teachers put on entertaining evenings in grade school gymnasiums where kids would play on big tarps illustrated with data structures. How cool is that? The kids loved it. I hope some of them were inspired as I was by this deeply educational involvement very concretely in very abstract work.

Mike Fellows shows the way to the popularization of computer science. And, Lord knows, it needs it. As an artist, I am continually struck by the near complete ignorance in the general culture of all matters pertaining to the theory of computation. And this goes very deeply into the fears people have of the role of machines in our lives. Mike blows all that fear away with his ability to engage curiosity and thought in play. Kids and

adults look at mathematics and computers very differently after one of these evenings. It is presented as something they can think about and deal with in a very playful way. Brilliant. Just brilliant, really.

Mike had a big influence on my vision of many things, such as the role of the theory of computation in digital art and poetics. Many artists think of computers as glorified televisions, or stereos, or typewriters. The radical flexibility of computers is invisible to them, so that they conceive of the vistas of digital art as, fundamentally, more of the same. But computing not only dissolves the borders between media, given that it is all coded in zeroes and ones, but it posits whole new media. Continually. Mike helped me see that the theory of computation is one of the philosophical underpinnings of any interesting philosophy of computer art. He has that sort of effect on people. He teaches very deeply. Not simply at the level of puzzle, but that of enigma and life-long involvement. Thank you, Mike. You are a cosmos and contain multitudes.

9.3 Educational Game Design

Mike and Frances met in September, 1998 (when Mike was presenting mathematical plays at the Victoria Fringe Festival, described in the chapter on Passion Plays). They quickly learned that they worked well together; sometimes Mike calls them the “Mike and Fran Team.” They became consultants for a start-up company in Texas that was designing a game to teach basic chemistry. They realized that there are key questions that must be addressed by the chemistry game or any game trying to be both educational and entertaining. Below are a few of the issues in their (unpublished) *Catalogue of Educational Computer Game Design Philosophy*.

1. *Chocolate-Covered Broccoli (CCB)* — sweet coating over bitter substance — is unpalatable to us. To produce a game with curriculum content that is not disguised as unpalatable CCB, a first plan is to analyze the overall structure of the mathematical curriculum, and the architecture of the games approach.
2. *Missed Magic* means throwing out the chocolate just because the recipe also needs a tie-in spice like nutmeg, a format that catalyzes a relationship between the bitter and the sweet elements. The real chocolate is wonder and curiosity. How does the game pay systematic attention to these fundamental resources, including the catalytic format that keeps the wonder going along with the learning? Mike recalled a period when his small child had a chemistry set with galvanizing images that kept her wonder going on a day-to-day basis. “The elements were her personal little friends,” Mike said, “the sprites of the world to which she had secret access through her knowledge of Chemistry and atoms.”
3. *Ageless Chocolate versus Perishable Broccoli*. Edutainment inherently has conflicting timescales: eternal chocolate versus perishable broccoli. How does one combine an authentic expression of the universal ageless intrinsic appeal of the subject, and the grade-specific curriculum agenda?

Mike and Frances posited that an educational game must possess self-awareness. A game just for entertainment has a simple position: pure fun. A game for education has a simple, authoritarian position: eat this broccoli. It is good for you. In contrast, an edutainment game must express some awareness of its predicament in trying to convey a majestic quest while also being a server of broccoli.

They used the insights they had gained from working on the chemistry game when they moved to New Zealand in 1999. There, they designed a multi-player, auction-type game for children that could be played on a mobile-phone. Mobile phones had become so ubiquitous, even among children, that they believed they now had a global catalyst for introducing mathematics. Mike and Frances were well coached by the Wellington Innovation Council, and eventually presented their business plan to a prominent Sydney venture capitalist. “Mr. B.” understood science (he had a Ph.D. in Chemical Engineering) and his heartstrings were sympathetic to their project. He thought he would buy their game for his daughter and that she would like it.

There were two ways that children could interact with the game. They could play by identifying a minimum vertex cover on a graph of dots and lines that was presented on their mobile. The second way children could join the game was to compete in creating graphs (on a given number of vertices) for which it was very hard to find a minimum vertex cover. Mike and Frances would collect and analyze these children-designed graphs, and award prizes for those judged the hardest. In this way, the children would be participating in cutting-edge scientific research. The children would be helping scientists come to understand what makes some graph problems *hard*. As Jim has said above, bringing children along on the grand venture into the unknown is Mike’s hallmark. Mike and Frances thought the scientific participation aspect of the game would increase its attractiveness to parents, as well as to the children.

Mike is emphatic that people understand that working with children is not just a good deed.

Mathematical scientists explaining science to kids is not some sort of noble, but career-nonsensical worthy cause, karma points, like volunteering for an NGO, or Scientists Without Borders. In reality, the effort to explain science to kids is a vital source of new mathematical insights for adults [in this volume, see Koblitz’ chapter about Kid Crypto yielding advances in the research area of Grobner Bases cryptosystems — Google on “Poly Cracker cryptosystems”.] The effort to communicate math to children is a real win-win.

Major fields do not often undergo major paradigm shifts, yet that is what must happen in education in order that classrooms communicate real mathematics. The issue of assessment and student evaluation has been a major impediment. While at VUW, Mike and Frances proposed a game-like assessment environment. Mike is cautiously confident that assessment along the lines of citation analysis engines (such as Harzing’s “Publish or Perish”) will speed the (inevitable) mathematical sciences education revolution.

Mike has created or supported many venues for children. The *Erdős for Kids* website, with child-sized open problems and prizes, *MEGA-Math!*, *Unplugged*, Family Nights, computer games, plays — all these and more are manifestations of Mike’s quest to bring open, unsolved problems in mathematics to the children — a sense of mathematics as a live, dynamic and wondrous enterprise.

Table 3. Overview of key events in the history of the Computer Science Unplugged project

1989	Mike Fellows and Nancy Casey meet in Idaho and develop activities that become <i>MEGA-Math</i>
1989	Tim Bell starts developing computer science exhibits for children for science center displays
1991	Mike writes his “manifesto”: Computer SCIENCE and Mathematics in the Elementary Schools [4]
1992	Mike and Nancy publish <i>This is MEGA-Mathematics!</i>
1992	Tim develops games and magic trick for classroom use
1993	Tim visits Mike in Victoria, BC for one month, and they plan the book that becomes CS Unplugged
1994	Mike speaks at the Games Development Conference
1995	Ian Witten joins the project to create a broader range of activities and help with writing
1997	Development of Mike’s “Cowboy Melodramas”
1998	Unplugged book has been rejected by 27 publishers, so is released as “shareware” on the Internet
2000	Revised version for teachers is released
2003	ACM K-12 curriculum released, recommending several Unplugged activities
2005	First Unplugged video is made when Bengt Aspvall visits Tim in NZ
2006	First translated book (Korean) is released
2006	Google supports the Unplugged project so the entire book can be released at no cost
2010	Chinese version (re-written for students) is released

10 Conclusion

Some of the key dates in the history of CS Unplugged and related work are summarized in Table 3. It is permeated with collaborations that appear to be initiated by chance meetings (e.g. Mike and Nancy met through their children being in the same class, Mike and Tim met through an internet discussion, and Mike and Fran met through the “Passion Plays”), although given Mike’s passion for this work, it was inevitable that he would attract collaborators who shared the vision and would work hard to see it bear fruit. These collaborations have each resulted in long-term productive relationships that have had a significant impact on computing education all around the world. CS Unplugged and related projects have engaged students, inspired teachers and empowered presenters to communicate the heart of mathematics and computer science without the

distraction of computers. They were driven by the authors' passion for communicating what math and computer science is all about to a public who misunderstood both fields, not by preaching to them about what it is, but by having them *do* math and computer science. They have grown thanks to input from open-minded students and educators who have embraced the ideas, and enthusiastic principals and leaders in education who have supported and promoted it.

Right from the start the broad vision for *MEGA-Math*, Unplugged and related projects has come from Mike, initially through his writings and advocacy in the early 1990s, and his continued creativity and demonstration of the ideas in many countries in the two decades since. A common thread that comes out in Mike's creative problem-solving style in everything he's done, from parameterized complexity to working with kids, is that it isn't about trying to prove his own ability, but the way he has cultivated excitement amongst those around him. He had been captivated by ideas, had come up with wonderfully creative ways to communicate them, and passed his passion on to anyone who would listen, whether colleagues, teachers or young children. It seems that the children were the quickest to embrace the ideas! He just didn't pay any attention to boundaries—for example, working with kids it never occurred to him that there should be any barriers to teaching mathematics as something fun and wonderful. Math as stand-up comedy? Why not! Computer science without computers? Of course! This attitude applies to his research—he seems to break new ground because it doesn't occur to him not to go there. He is just as passionate talking about math and computer science to 6-year-old children or experienced professors.

Who knows what the future holds for Unplugged; computer science as a subject is starting to appear in schools around the world, and Unplugged may find its way into school text books. If virtual worlds become popular in teaching, Unplugged already has entered that culture. New activities may well appear, although it seems that there is more interest in adaptation of existing activities, including translations, videos, followup ideas, and providing more detailed background information.

Some time around 2010 someone on a newsgroup commented on how quickly computing books go out of date, asking “What 14 year old computing book would you ever want to use?” It was refreshing to realize at the time that the Unplugged book was written about 14 years earlier, and many of the activities were virtually unchanged from the version created in the early 1990s. Even now there's no urgency to update the ideas as they reflect timeless fundamentals of the subject, and are still enjoyed by young and old in many cultures. A recurring theme of Mike's work is a sense of story and drama, and these won't date; in the last 20 years we have seen software companies come and go, and hardware become obsolete many times over, but children—and adults—remain captivated by the stories that bring math and computer science to life.

In the end, this reminds us that computing is about people and not computers. We give the last word to a computing teacher in Japan, who commented on her experience using Unplugged:

“Now the teacher sees the children’s faces instead of the back of the computers” (Yayoi Hofuku, teacher at Shouyou High School, Tokyo, Japan).

Acknowledgements. We are grateful to the following people who have contributed their ideas and checked details for this article: Jim Andrews, Bengt Aspvall, Judith Bell, Lenore Blum, Neal Koblitz, Rita Liff Levinson, Geri Lorway, Sumant Murugesh, Heidi Newton, Lenny Pitt, Matt Powell, Ulrike Stege, Geoff and Lisa Whittle, and Ian Witten.

The Unplugged project has benefited from the work of countless people over the years, many of whom are named in this chapter, but others whose great ideas were picked up in passing and the source has been lost. To all those people who embraced the vision we express our gratitude.

The main advisors for the Unplugged project after it received formal funding in 2006 have been Bengt Aspvall, Lenore Blum, Anna Charny, Sam Chung, Tom Cortina, Robb Cutler, Peter J Denning, Rick Dipaolo, Paul Gibson, Pam Hagen, Mindy Hart, Peter Henderson, Susumu Kanemune, Rachel Kestenbaum, Lynn Lambert, Lee Won Gyu, Geri Lorway, Craig Nevill-Manning, Andy and Todd Seymour, Harold Thimbleby, Alfred Thompson, Allen Tucker, David Vogt, Len Wanger, Xia Xie, and Ben Tsutom Wada. Significant support for Unplugged has been provided by Jeff Walz, the University relations contact at Google.

The *MEGA-Math* and Unplugged books acknowledge many teachers and colleagues who supported these projects over the years, but a key role was played by the authors’ children, Andrew, Anna, Elizabeth, Hannah, Max, Michael, Nikki and Patrick who “inspired much of this work, and were often the first children to test an activity.” They have been involved for around 20 years, and our children have now grown up, but Unplugged and the associated work will no doubt continue to benefit from real research being done by young children all over the world!

References

1. Garey, M.R., Johnson, D.S.: Computers and intractability: a guide to the theory of NP-completeness. W. H. Freeman, San Francisco (1979)
2. Bell, T., Marghitu, D., Bell, J.: Workshop: CS Unplugged — computational thinking without computers. In: Pirkul, H., Spong, M.W., Shah, R., Suh, S. (eds.) Proceedings of SDPS 2011. Society for Design and Process Science, Jeju Island (2011)
3. Rosamond, F.: On-line and off-line computer games and mathematical sciences popularization. In: Rosamond, F., Copes, L. (eds.) Educational Transformations: The Influences of Stephen I. Brown, pp. 407–426. Authorhouse, Bloomington (2006)
4. Fellows, M.: Computer SCIENCE in the elementary schools. In: Fisher, N., Keynes, H., Wagreich, P. (eds.) Proceedings of the Mathematicians and Education Reform Workshop of Issues in Mathematics Education. Conference Board of the Mathematical Sciences, Seattle, vol. 3, pp. 143–163 (1991)
5. Casey, N.: Megamath: Expanding and connecting the mathematics community. In: INET 1995, Internet Society’s 1995 International Networking Conference, Honolulu, Hawaii (1995)

6. Casey, N.: Whole language: lessons for math teachers (1990), <http://www.ccs3.lanl.gov/mega-math/papers/firest.ps>
7. Casey, N.: Three for the money: an hour in the classroom. In: Rosenstein, J.G., Franzblau, D.S., Roberts, F.S. (eds.) *Discrete Mathematics in the Schools*. DIMACS: Series in Discrete Mathematics and Theoretical Computer Science, A co-publication of the AMS, DIMACS, and National Council of Teachers of Mathematics (1997)
8. Casey, N., Fellows, M.R.: This is Mega-Mathematics! Los Alamos National Labs (1992), <http://www.ccs3.lanl.gov/mega-math/write.html>
9. Kierstead, H.A., Trotter, W.T.: Planar graph coloring with an uncooperative partner. *J. Graph Theory* 18, 569–584 (1994)
10. Casey, N., Fellows, M.: Implementing the standards: Let's focus on the first four. In: *How Can We Have an Impact?* DIMACS Series: Discrete Mathematics in the Schools (1997)
11. Fellows, M.: Research on mega-math: Discrete mathematics and computer science for children, final report (1995), <http://www.osti.gov/bridge/purl.cover.jsp?purl=/106599-70WLSu/webviewable/>
12. Hodder, P.: Science as theatre: a New Zealand history of performances and exhibitions. *Journal of Science Communication* 10, 1–10 (2011)
13. Bell, T.C.: Computer science for the uninterested: designing displays for a science centre. *Computers in New Zealand Schools* 4, 40–46 (1992)
14. Fellows, M.R., Koblitz, N.: Kid Krypto. In: Brickell, E.F. (ed.) *CRYPTO 1992*. LNCS, vol. 740, pp. 371–389. Springer, Heidelberg (1993)
15. Harel, D.: *Algorithmics: The Spirit of Computing*, 1st edn. Addison-Wesley, Reading (1987)
16. Vöcking, B., Alt, H., Dietzfelbinger, M., Reischuk, R., Scheideler, C., Vollmer, H., Wagner, D. (eds.): *Algorithms Unplugged*. Springer (2011)
17. Dewdney, A.K.: *The Turing omnibus: 61 excursions in computer science*. Computer Science Press, Rockville (1989)
18. Bell, T., Witten, I., Fellows, M.: *Computer Science Unplugged: Off-line activities and games for all ages (original book)* (1999), <http://csunplugged.org>
19. Bell, T., Witten, I., Fellows, M., McKenzie, J., Adams, R.: *Computer Science Unplugged: An enrichment and extension programme for primary-aged children* (2002), <http://csunplugged.org>
20. Bell, T., Thimbleby, H., Fellows, M., Witten, I., Koblitz, N., Powell, M.: Explaining cryptographic systems to the general public. *Computers and Education* 40, 199–215 (2003)
21. Bell, T.: A low-cost high-impact computer science show for family audiences. In: *Australasian Computer Science Conference 2000 (ACSC 2000)*, Canberra, Australia, January 31- February 3, pp. 10–16 (2000)
22. Tucker, A., Deek, F., Jones, J., McCowan, D., Stephenson, C., Verno, A.: *A Model Curriculum for K-12 Computer Science: Final Report of the ACM K-12 Task Force Curriculum Committee*. ACM, New York (2003)
23. Bell, T., Witten, I., Fellows, M., Lee, W., McKenzie, J., Adams, R.: *Computer Science Unplugged: Off-line activities and games for all ages*. Hongreung Science Publishing, Seoul (2006) (in Korean)
24. Blum, L., Cortina, T.J.: CS4HS: an outreach program for high school CS teachers. In: Russell, I., Haller, S.M., Dougherty, J.D., Rodger, S.H. (eds.) *Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education, SIGCSE 2007*, Covington, Kentucky, USA, pp. 19–23. ACM (2007)

25. Blum, L., Cortina, T.J., Lazowska, E.D., Wise, J.: The expansion of CS4HS: an outreach program for high school teachers. In: Dougherty, J.D., Rodger, S.H., Fitzgerald, S., Guzdial, M. (eds.) Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education, SIGCSE 2008, Portland, OR, USA, pp. 377–378. ACM (2008)
26. Bell, T., Bell, R.: The Treasure Hunt. In: Fitzgerald, S., Guzdial, M., Lewandowski, G., Wolfman, S.A. (eds.) Proceedings of the 40th ACM Technical Symposium on Computer Science Education. ACM, New York (2009)
27. Bell, T., Bell, R.: The Orange Game. In: Fitzgerald, S., Guzdial, M., Lewandowski, G., Wolfman, S.A. (eds.) Proceedings of the 40th ACM Technical Symposium on Computer Science Education, SIGCSE 2009. ACM, New York (2009)
28. Bell, T., Bell, R.: Image Compression - Making Contact. In: Lewandowski, G., Wolfman, S.A., Cortina, T., Walker, E. (eds.) SIGCSE, Milwaukee, WI. ACM (2010)
29. Bell, T., Bell, R.: Sorting Algorithms (Unplugged). In: Lewandowski, G., Wolfman, S.A., Cortina, T., Walker, E. (eds.) SIGCSE, Milwaukee, WI. ACM (2010)
30. Bell, T., Bell, M., Finnerty, K.: Computer Science Buskers (Error Correction). In: Lewandowski, G., Wolfman, S.A., Cortina, T., Walker, E. (eds.) SIGCSE, Milwaukee, WI. ACM (2010)
31. Bell, T., Bell, M., Chicha, V.: Santa's dirty socks (divide and conquer). In: Cortina, T.J., Walker, E.L., King, L.S., Musicant, D.R. (eds.) Proceedings of the 42nd ACM Technical Symposium on Computer Science Education, SIGCSE 2011. ACM, New York (2011)
32. Bell, T., Bell, M.: Reaching out (binary codes). In: Proceedings of the 42nd ACM Technical Symposium on Computer Science Education, SIGCSE 2011. ACM, New York (2011)
33. Sagan, C.: Contact. Simon and Schuster, New York (1985)
34. Yoo, S., Yeum, Y., Kim, Y., Cha, S., Kim, J., Jang, H., Choi, S., Lee, H., Kwon, D., Han, H., Shin, E., Song, J., Park, J., Lee, W.: Development of an Integrated Informatics Curriculum for K-12 in Korea. In: Mittermeir, R.T. (ed.) ISSEP 2006. LNCS, vol. 4226, pp. 199–208. Springer, Heidelberg (2006)
35. Choi, S.K., Bell, T., Jun, S.J., Lee, W.G.: Designing offline computer science activities for the Korean elementary school curriculum. In: ITiCSE 2008: Proceedings of the 13th annual conference on Innovation and Technology in Computer Science Education, p. 338. ACM, New York (2008)
36. Bell, T.C., Witten, I.H., Fellows, M.R.F., Kanemune, S., Kuno, Y.: Computer Science Unplugged: Off-line activities and games for all ages, Etext, Tokyo, Japan (2007) (in Japanese)
37. Fellows, M., Hibner, A., Koblitz, N.: Cultural aspects of mathematics education reform. Notices of the American Mathematics Society 41, 5–9 (1994)
38. Bell, T., Wada, B.T., Kanemune, S., Xie, X., Lee, W., SookKyoung, C., Aspvall, B., Wingkvist, A.: Making computer science activities accessible for the languages and cultures of Japan, Korea, China and Sweden. In: Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education, SIGCSE 2008, Portland, OR, USA (2008)
39. Bell, T., Arpaci-Dusseau, A., Witten, I., Fellows, M.: Computer Science Unplugged: understanding computing through games and puzzles. HUST Press, Wuhan (2010)

40. Goode, J.: Connecting K-16 curriculum and policy: making computer science engaging, accessible, and hospitable for underrepresented students. In: Lewandowski, G., Wolfman, S.A., Cortina, T.J., Walker, E.L. (eds.) Proceedings of the 41st ACM Technical Symposium on Computer Science Education, SIGCSE 2010, Milwaukee, Wisconsin, USA, March 10-13, pp. 22–26. ACM (2010)
41. Goode, J., Margolis, J.: Exploring computer science: A case study of school reform. *Trans. Comput. Educ.* 11, 12:1–12:16 (2011)
42. Denning, P.J.: Computing is a natural science. *Commun. ACM* 50, 13–18 (2007)
43. Wing, J.M.: Computational thinking. *Commun. ACM* 49, 33–35 (2006)
44. National Research Council: Report of a Workshop on the Scope and Nature of Computational Thinking (2010), http://www.nap.edu/catalog.php?record_id=12840
45. Marghitu, D., Fuller, M., Brahim, T.B., Banu, E.: Auburn university robotics and computer literacy K-12 engineering camps: A success story. In: Bernal, B. (ed.) ASEE-SE 2009, GA USA, April 5-7. Southern Polytechnic State University, Marietta (2009)
46. Ward, B., Bell, T., Marghitu, D., Lambert, L.: Teaching computer science concepts in Scratch and Alice. *The Journal of Computing Sciences in Colleges* 26, 173–180 (2010)
47. Thompson, D., Bell, T.: Virtual worlds: Evaluating CS education activities through automated monitoring. In: Pirkul, H., Spong, M.W., Shah, R., Suh, S. (eds.) Society for Design and Process Science, SDPS 2010, Dallas, Texas, pp. 1–7 (2010)
48. Voigt, J., Bell, T., Aspvall, B.: Competition-style programming problems for computer science unplugged activities. In: Verdu, E., Lorenzo, R., Revilla, M., Regueras, L. (eds.) A New Learning Paradigm: Competition Supported by Technology, CEDETEL, Boecillo, Spain, pp. 207–234 (2009)
49. Magnusson, W.E., Mourão, G.: *Statistics without Math*. Sinauer Associates (2004)
50. Koblitz, N.: The case against computers in K-13 math education (kindergarten through calculus). *The Mathematical Intelligencer* 18, 9–16 (1996)
51. Fellows, M., Parberry, I.: SIGACT trying to get children excited about CS. *Computing Research News* 7 (1993)
52. Nishida, T., Kanemune, S., Idosaka, Y., Namiki, M., Bell, T., Kuno, Y.: A CS Unplugged design pattern. In: Proceedings of the 40th SIGCSE Technical Symposium on Computer Science Education, SIGCSE 2009, Chattanooga, TN, USA, pp. 231–235. ACM, New York (2009)
53. Taub, R., Ben-Ari, M., Armoni, M.: The effect of CS unplugged on middle-school students' views of CS. In: Brézillon, P., Russell, I., Labat, J.M. (eds.) Proceedings of the 14th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education, ITiCSE 2009, Paris, France, July 6-9, pp. 99–103. ACM (2009)
54. Feaster, Y., Segars, L., Wahba, S.K., Hallstrom, J.O.: Teaching CS unplugged in the high school (with limited success). In: Rößling, G., Naps, T.L., Spannagel, C. (eds.) Proceedings of the 16th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education, ITiCSE 2011, Darmstadt, Germany, June 27-29, pp. 248–252. ACM (2011)
55. Carmichael, G.: Girls, computer science, and games. *SIGCSE Bull* 40, 107–110 (2008)
56. Hart, M., Early, J.P., Brylow, D.: A novel approach to K-12 CS education: linking mathematics and computer science. In: Dougherty, J.D., Rodger, S.H., Fitzgerald, S., Guzdial, M. (eds.) Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education, SIGCSE 2008, Portland, OR, USA, pp. 286–290. ACM (2008)

57. Lambert, L., Guiffre, H.: Computer science outreach in an elementary school. *J. Comput. Small Coll.* 24, 118–124 (2009)
58. Groover, T.R.: Using games to introduce middle school girls to computer science. *J. Comput. Small Coll.* 24, 132–138 (2009)
59. Curzon, P., McOwan, P.W.: Engaging with computer science through magic shows. In: *Proceedings of the 13th Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE 2008*, pp. 179–183. ACM, New York (2008)
60. Cottam, J.A., Foley, S.S., Menzel, S.: Do roadshows work?: examining the effectiveness of just be. In: Lewandowski, G., Wolfman, S.A., Cortina, T.J., Walker, E.L. (eds.) *Proceedings of the 41st ACM Technical Symposium on Computer Science Education, SIGCSE 2010*, Milwaukee, Wisconsin, USA, March 10–13, pp. 17–21. ACM (2010)
61. Morreale, P., Joiner, D.: Reaching future computer scientists. *Commun. ACM* 54, 121–124 (2011)
62. Hart, M.L.: Making contact with the forgotten K-12 influence: are you smarter than your 5th grader? In: *Proceedings of the 41st ACM Technical Symposium on Computer Science Education, SIGCSE 2010*, pp. 254–258. ACM, New York (2010)
63. Carruthers, S.: Grasping graphs. Master of Science thesis, University of Victoria, Victoria, BC (2010), <http://hdl.handle.net/1828/3193>
64. Carruthers, S., Milford, T., Pelton, T., Stege, U.: Draw a social network. In: *Proceedings of the 16th Annual Joint Conference on Innovation and Technology in Computer Science Education, ITiCSE 2011*, pp. 178–182. ACM, New York (2011)
65. Bell, T., Curzon, P., Cutts, Q., Dagiene, V., Haberman, B.: Overcoming Obstacles to CS Education by Using Non-programming Outreach Programmes. In: Kalaš, I., Mittermeir, R.T. (eds.) *ISSEP 2011*. LNCS, vol. 7013, pp. 71–81. Springer, Heidelberg (2011)
66. Fellows, M.: The heart of puzzling: Mathematics and computer games. In: *Proceedings of the 1996 Computer Games Developers Conference*, pp. 109–120. Miller Freeman (1996)

FPT Suspects and Tough Customers: Open Problems of Downey and Fellows

Fedor V. Fomin¹ and Dániel Marx²

¹ Department of Informatics
University of Bergen, Norway

² Computer and Automation Research Institute
Hungarian Academy of Sciences (MTA SZTAKI)
Budapest, Hungary

Abstract. We give an update on the status of open problems from the book “Parameterized Complexity” by Downey and Fellows.

1 Introduction

Downey and Fellows’ 1999 monograph [14] contains a list of open problems which strongly influenced the development of Parameterized Complexity in the following decade. Here we survey the current status of these problems.

Downey and Fellows partitioned their list of problems into two parts: “A Lineup of FPT Suspects” and “A Lineup of Tough Customers.” While within the time some of the FPT suspects appeared to be tough customers and, vice versa, some of the tough customers turned to be not that tough, in our survey we decided to keep the original order and partition.

We do not provide definitions of classes FPT, XP, and W-hierarchy, referring to the book of Downey and Fellows [14], as well to more recent monographs of Flum and Grohe [18], and Niedermeier [37].

It is worthwhile to look back on this list now, more than 10 years later of its publication, and to try to see what we can learn from its history. An immediate and somewhat surprising observation is that with the exception of two problems, all the questions were resolved in the positive by fixed-parameter tractability results, even many of those which were classified as “tough customers” by Downey and Fellows. One can say that the algorithmic side of fixed-parameter tractability developed much more dramatically since 1999 than the complexity side. In the past 10 years, several fundamental and powerful techniques were introduced into the positive toolkit of fixed-parameter tractability (e.g., bidimensionality, iterative compression, algebraic techniques, inclusion-exclusion, various forms of randomization, etc.). On the other hand, while $W[1]$ -hardness proofs got more streamlined over the years and we have now a better understanding of how to obtain hardness results for certain types of problems (e.g., for planar or bounded-treewidth problems), we do not have such a richness of standard techniques as in the case of algorithmic results. For most $W[1]$ -hardness proofs, we still have to roll up our sleeves and reduce from MAXIMUM CLIQUE by constructing appropriate gadgets. If this trend continues, then we can expect to see further

exciting developments in parameterized algorithmic techniques for several years. Apparently, the tools and theory of fixed-parameter tractability are even more deep and diverse than what Downey and Fellows expected in 1999 (and possibly what we see now). It is conceivable that in many cases, the main roadblock to understanding the complexity of a problem is not our limited ability to do $W[1]$ -hardness proofs, but the fact that the right algorithmic technique for the problem is still waiting to be discovered.

2 A Lineup of FPT Suspects

TOPOLOGICAL CONTAINMENT

FPT

Instance: An undirected graph G

Parameter: A graph H

Question: Is H topologically contained in G ?

Graph H is topologically contained in G if a subdivision of H is a subgraph of G . The problem is in XP because one can guess all possible mapping of vertices of H into G and then for each guess apply the disjoint path algorithm of Robertson and Seymour [39]. The problem was shown to be in FPT by Grohe, Kawarabayashi, Marx, and Wollan in 2011 [23]. For every fixed undirected graph H , they gave an $O(|V(G)|^3)$ time algorithm for testing if a given graph G contains H topologically.

IMMERSION ORDER TEST

FPT

Instance: An undirected graph G

Parameter: A graph H

Question: Does H has an immersion in G ?

An immersion of graph H into graph G is a mapping of vertices of H into vertices of G such that edges of H correspond to edge-disjoint paths of G . The problem is in FPT and solvable in $O(|V(G)|^3)$ time by reduction to TOPOLOGICAL CONTAINMENT [23].

DIRECTED FEEDBACK VERTEX SET

FPT

Instance: A directed graph G

Parameter: A positive integer k

Question: Is there a set S of k vertices such that each directed cycle of G contains a member of S ?

The problem was shown to be in FPT by Chen, Liu, Lu, O’Sullivan, and Razgon in 2008 [6,7]. The running time of the algorithm is $4^k k! n^{O(1)}$. It remains open if there exist a single exponential algorithm for DIRECTED FEEDBACK VERTEX SET even on planar graphs. The existence of polynomial kernel is also open. The undirected variant of the problem, FEEDBACK VERTEX SET, received much more attention: the problem was proved to be in FPT by a simple combinatorial algorithm already in [13], it is known to be solvable in single exponential time [10,25], and admits a polynomial kernel [43].

PLANAR DIRECTED DISJOINT PATHS	Open
Instance: A directed planar graph G and k pairs $\langle r_1, s_1 \rangle, \dots, \langle r_k, s_k \rangle$ of vertices of G	
Parameter: k	
Question: Does G have k vertex-disjoint paths P_1, \dots, P_k with P_i running from r_i to s_i ?	

The problem is open. Problem is in XP: Schrijver [41] showed that the problem is polynomial-time solvable for every fixed k . We remark that the paper of Schrijver is self-contained and in particular it does not use results from Graph Minors theory. The NP-hardness of the problem follows from the fact that even the undirected problem is NP-hard on planar graphs. For general graphs, the directed problem is NP-hard already for $k = 2$ [19].

PLANAR t-NORMALIZED WEIGHTED SATISFIABILITY	FPT
Instance: A planar t -normalized formula X	
Parameter: A positive integer k	
Question: Does X have a satisfying assignment of weight k ?	

A Boolean formula is t -normalized if it is of the form $\bigwedge \bigvee \bigwedge \dots$ of literals with $t-1$ alternations of the \bigwedge and \bigvee quantifiers. For example, a 2-normalized formula is a CNF formula.

A CNF formula is planar if the bipartite graph of the formula (where one class is the set of clauses, the other class is the set of variables) is planar. However, it is not clear what the definition of a planar t -normalized formula should be and it is not defined in [14]. One obvious definition could be that the Boolean circuit describing the formula is planar. The problem with this definition is that “planar CNF formula” and “planar 2-normalized formula” are two different notions: the latter variant is more restrictive, as the Boolean circuit contains an output gate that is connected to all clauses. This suggests another, less restrictive, definition: a t -normalized formula is planar if the Boolean circuit describing the formula *with the output gate removed* is planar.

The problem is FPT even with the less restrictive definition of planarity. This follows from the fact that, for every fixed k and t , there is a first-order formula (over an appropriate planar structure) that expresses the existence of a weight- k satisfying assignment. Therefore, a powerful general result of Frick and Grohe [20] implies a linear-time algorithm for every fixed k and t . To construct this formula, one needs to express that there exists k variables such that the output gate (or more precisely, every input of the output \wedge gate) is satisfied. As the formula is t -normalized, at most t quantifiers are needed to express that a gate is satisfied.

We sketch how a direct solution can be obtained by the standard layering and bounded-treewidth techniques on planar graphs (“Baker’s shifting strategy”). The all-zero assignment determines a “standard” value v_g for every gate g . The key observation is that the only way g can have the opposite of v_g in some assignment if g is at distance at most t from a variable with value 1. This means that if we partition the graph into layers, then in every assignment of weight k , all but at most $(2t+1)k$ layers have the property that every gate has the standard value. By starting at some layer $i \leq (2t+1)k$ and forcing every $(2t+1)k+1$ -st layer to take the standard value, the problem falls apart into independent subproblems, each having at most $(2t+1)k$ layers. Graphs with bounded number of layers are known to have bounded treewidth, hence the subproblems can be solved using standard techniques. Finally, our observation above implies that if there is a solution, then at least one choice of starting layer i is consistent with this solution, hence our algorithm finds a solution when considering this choice of i .

PLANAR MULTIWAY CUT

W[1]-hard

Instance: A weighted undirected planar graph G with terminals $\{x_1, \dots, x_k\}$ and a positive integer M

Parameter: k

Question: Is there a set of edges of total weight $\leq M$ whose removal disconnects each terminal from all others?

The problem is known to be in XP: it can be solved in time $n^{O(k)}$ [27,9] and more recently in time $2^{O(k)} \cdot n^{O(\sqrt{k})}$ [31]. The problem was shown to be W[1]-hard in 2011 [34]. Furthermore, assuming the Exponential Time Hypothesis [28], there is no $f(k)n^{o(\sqrt{k})}$ time algorithm for the problem.

For general graphs, the problem is NP-hard already for $k = 3$ [9]. When parameterized by the total weight of the solution, the problem is FPT on general graphs [35,5,24,8] (the number of terminals can be arbitrary). The vertex-removal variant where the parameter is the total weight of the vertices to be deleted is also in FPT: the most recent algorithm of Cygan et al. [8] achieves the same running time for both versions.

3 A Lineup of Tough Customers

FIXED ALPHABET LONGEST COMMON SUBSEQUENCE (LCS) W[1]-hard

Instance: k sequences X_i over an alphabet Σ of fixed size and a positive integer m

Parameter: k

Question: Is there a string $X \in \Sigma^*$ of length m that is a subsequence of each of the X_i ?

Note that the characters in the subsequence X need not be consecutive in X_i . A simple $O(n^{k+1})$ time dynamic programming algorithm shows that the problem is in XP. When the size of the alphabet Σ is not bounded or when the parameter is $k + |\Sigma|$, the problem was known to be W[t]-hard for every $t \geq 1$ already in 1995 [3,4]. Pietrzak in 2003 [38] showed that the problem is W[1]-hard parameterized by k , even if the alphabet is binary.

BOUNDED HAMMING WEIGHT DISCRETE LOGARITHM

Open

Instance: An n -bit prime p , a generator g of F_p^* , an element $a \in F_p^*$

Parameter: A positive integer k

Question: Is there a positive integer x whose binary representation has at most k 1's (that is, x has a Hamming weight of k) such that $a = g^x$?

Here F_p^* is the multiplicative group of non-zero integers modulo p . Element $g \in F_p^*$ is a generator of group F_p^* if for every element a , there exists an integer x with $a = g^x$. Note that there is a unique $1 \leq x \leq p - 1$ with $a = g^x$, but the problem definition does not insist that x should be less than p . To show that the problem is in XP, we need to argue that the representation of x is at most kn bits long (hence there are at most $(kn)^k$ different possibilities for x to try). See [17] for discussion and related problems.

The famous DISCRETE LOGARITHM problem is to find the unique $1 \leq x \leq p - 1$ with $a = g^x$; the hardness of some cryptosystems are based on the assumed hardness of this problem. Because the problem definition does not require $x \leq p$, it is not completely obvious how the two problems relate to each other.

CROSSING NUMBER

FPT

Instance: An undirected graph G

Parameter: A positive integer k

Question: Is the crossing number of G is at most k ?

The crossing number of a graph is the minimum number of edge crossings in a planar drawing of the graph (with the usual technical assumptions, such as no

three edges cross at the same point). A graph is a planar graph if and only if its crossing number is 0. The problem asks if G can be drawn with at most k edge crossings. The problem was solved by Grohe in 2001 [22,21], who showed that the problem is solvable in time $O(|V(G)|)^2$ for every fixed k . A linear-time algorithm is claimed in [30].

Downey and Fellows formulate the CROSSING NUMBER problem as “Can G be embedded in the plane with at most k edges crossing?”, which can be interpreted as finding an embedding in which at most k edges participate in crossings. This is different from the classical definition of crossing number, but could be an interesting problem on its own right. A related problem is deciding if a graph is in the class “Planar+ ke ”, meaning that it can be made planar by removing at most k edges. A linear-time algorithm is claimed also for this problem by Kawarabayashi and Reed [30]. Note that having at most k edges participating in crossings and removing k edges to make the graph planar are two different problems: if a graph has an embedding where k edges participate in crossings, then the graph can be made planar by removing less than k edges (as there is no need to remove all the edges participating in crossings).

MINIMUM DEGREE GRAPH PARTITION

FPT

Instance: An undirected graph G

Parameter: Positive integers k and d

Question: Can $V(G)$ be partitioned into disjoint subsets V_1, \dots, V_m so that for $1 \leq i \leq m$, $|V_i| \leq k$ and at most d edges have exactly one endpoint in V_i ?

Langston and Plaut [32] observed in 1998 that the graphs having such partitions for a fixed k and d are closed under taking immersions. Robertson and Seymour [40] proved that immersion is a well-quasi-ordering, which means that classes of graphs closed under immersion can be characterized by a finite number of forbidden immersed graphs. Together with the fact that the disjoint path algorithm of Robertson and Seymour [39] implies, for every fixed H , a polynomial-time algorithm for testing if H is immersed in G , it follows that the problem is in XP jointly parameterized by k and d . The result in 2011 that immersion testing is FPT [23] immediately implies that MINIMUM DEGREE GRAPH PARTITION is in (nonuniform) FPT.

Lokshtanov and Marx [33] showed in 2011 that the problem is in FPT parameterized by k or by d by establishing a more general result. In the (μ, p, q) -PARTITION problem, the task is to find a partition of the vertices where each cluster C satisfies the requirements that at most q edges leave C and $\mu(C) \leq p$. It was shown in [33] that when μ is one of the following functions—number of nonedges in the cluster, maximum degree of nonedges in the cluster, number of vertices in the cluster— (μ, p, q) -PARTITION can be solved in time $2^{O(p)}n^{O(1)}$ and in time $2^{O(q)}n^{O(1)}$, i.e., the problem is fixed-parameter tractable parameterized by p or by q .

SHORT CHEAP TOUR	FPT
Instance: A graph G , integer S , and edge weighting $w: E(G) \rightarrow \mathbb{Z}$	
Parameter: A positive integer k	
Question: Is there a tour through at least k nodes of G of cost at most S ?	

As observed by Fellows [16] in 2001, the problem is FPT by a simple reduction to finding a minimum weight cycle of length exactly k , which can be solved by color coding [2]. We sketch the reduction. Let G' be a complete graph on the same set of vertices as G , and let the weight of edge uv be the length of the shortest path between u and v in G . It is easy to see that G has a tour visiting at least k nodes of cost at most S if and only if G' has a cycle of length *exactly* k of cost at most S .

The variant of the problem where we ask that the cost of the tour is exactly S is W[1]-hard [12].

POLYMATROID RECOGNITION	Open
Instance: A k -polymatroid M	
Parameter: A positive integer k	
Question: Is M hypergraphic?	

Let E be a finite set. A polymatroid is a function $\rho : 2^E \rightarrow \mathbb{Z}$ with the following properties:

1. $\rho(\emptyset) = 0$,
2. $\rho(A) \leq \rho(B)$ for every $A \subseteq B \subseteq E$, and
3. $\rho(A) + \rho(B) \geq \rho(A \cap B) + \rho(A \cup B)$.

A k -polymatroid is a polymatroid with $\rho(e) \leq k$ for every $e \in E$. Given a hypergraph H with vertex set V and edge set E , the hypergraphic polymatroid of H is a function $\chi_H : 2^E \rightarrow \mathbb{Z}$ defined by

$$\chi_H(A) = |\overline{A}| - \kappa(H|A),$$

where \overline{A} is the set of vertices contained in the edge set A , and $\kappa(H|A)$ is the number of components of the hypergraph H restricted to A (see [45] for more details). A polymatroid is hypergraphic, if it is the hypergraphic polymatroid of a hypergraph.

A word of caution should be said on how the polymatroid is given in the input. One possibility is that it is given by an oracle, but then the problem does not fit the framework of complexity theory defined by problems as languages (but it is still an interesting question if $f(k) \cdot n^{O(1)}$ oracle calls are sufficient for the problem).

CHAIN MINOR ORDERING

Open

Instance: A finite poset Q **Parameter:** A finite poset P **Question:** Is P a chain minor of Q ?

Let $P = (V, <)$ be a poset. A chain is a sequence of elements $x_1 < x_2 < \dots < x_n$. We say that $P = (V, <)$ is a chain minor $P' = (V', <)$ if there is a partial mapping $\rho : V' \rightarrow V$ with the following property: for every chain C of P , there is a chain C' of P' such that ρ restricted to C' is an isomorphism of chains from C' to C . Gustedt [26] showed that the problem is in XP and that the chain minor relation is a well-quasi-ordering. The problem remains open.

SHORT GENERALIZED HEX

Open

Instance: An undirected graph G with two distinguished vertices v_1 and v_2 **Parameter:** A positive integer k **Question:** Does player one have a winning strategy of at most k moves in Generalized Hex?

In Generalized Hex two players play on a graph with white and black pebbles. Player one plays with white and player two with black pebbles. Player one starts by placing a white pebble on a vertex of G . Then alternately players make moves, at each move a pebble is placed on an occupied vertex. Player one wins if he can construct a path of white vertices from v_1 to v_2 .

To the best of our knowledge, the problem remains open. Downey and Fellows [14] proposed that the problem is a good candidate for AW[*]-completeness. Towards this goal, Allan [42] showed that the problem is in AW[*].

JUMP NUMBER

FPT

Instance: A poset P **Parameter:** A positive integer k **Question:** Is the jump number of P at most k ?

Given a finite partially ordered set (or poset) $P = (V, <_P)$, let $L = (V, <_L)$ be a linear extension of P , that is a total order on the same ground set V of P , such that each couple of elements $u, v \in V$ for which $u <_P v$ implies $u <_L v$. A consecutive pair (v_i, v_{i+1}) of elements in L is a jump or setup of L if $v_i \not<_P v_{i+1}$. The jump number of P is the minimum number of jumps in L , where minimum is taken over all the linear extensions L of P .

The problem was shown to be in XP by El-Zahar and Schmerl [15] in 1984. McCartin showed in 2001 [36] that the problem is in FPT.

POLYNOMIAL PRODUCT IDENTITY

Open

Instance: Two sets of k multivariate polynomials p_i and q_i for $i = 1, \dots, k$

Parameter: k

Question: Does the following identity hold?

$$\prod_{i=1}^k p_i = \prod_{i=1}^k q_i?$$

The polynomials in the input are given by listing the monomials with nonzero coefficients. Note that there is no bound on the number of variables or on the degree of the polynomials. By multiplying out each product, we get at most n^k monomials and we can compare the two sides to test for equality. Therefore the problem is in XP.

As discussed in [29, Section 4.3], the Schwartz-Zippel Lemma provides a way of solving the problem in randomized polynomial time and therefore it is in randomized FPT. Thus the problem is unlikely to be W[1]-hard. It could still be a nontrivial question if the problem is in deterministic FPT. Answering this question may tell us something interesting about the tradeoff between randomness and running time.

SHORTEST VECTOR

Open

Instance: A basis $X = \{x_1, x_2, \dots, x_n\} \subset \mathbb{Z}^n$ for a lattice \mathcal{L}

Parameter: A positive integer k

Question: Is there a non-zero vector $x \in \mathcal{L}$, such that $\|x\|^2 \leq k$?

Here $\|x\|$ denotes the Euclidean (ℓ_2) norm of $x = (a_1, \dots, a_n)$, defined as $\sqrt{\sum_{i=1}^n a_i^2}$. The problem was shown to be NP-hard under randomized reduction by Ajtai in 1998 [1], settling a longstanding open problem. The problem is in XP: every vector x with $\|x\|^2 \leq k$ contains at most k nonzero coordinates. It could be interesting to investigate the problem for other ℓ_p norms as well.

EVEN SET

Open

Instance: An undirected red/blue bipartite graph $G = (\mathcal{R}, \mathcal{B}, E)$

Parameter: A positive integer k

Question: Is there a non-empty set of at most k vertices $R \subseteq \mathcal{R}$, such that each member of \mathcal{B} has an even number of neighbors in R ?

Open. The exact version of the problem, where $|R| = k$, is W[1]-hard [11]. Vardy [44] proved the NP-completeness of the problem in 1997, settling a longstanding

open problem. There are other equivalent ways of stating the problem, showing that this problem appears naturally in many contexts:

- Given a hypergraph H , is there a nonempty set S of at most k vertices, such that $|e \cap S|$ is even for every hyperedge e ?
- Given a matrix A over the two-element field $GF[2]$, is there a nonzero vector \mathbf{x} having at most k nonzero coordinates and satisfying $A\mathbf{x} = 0$?
- Given a binary linear code defined by a matrix A over $GF[2]$, are there two codewords with Hamming-distance at most k ?
- Given a binary matroid represented by a matrix A over $GF[2]$, does it have a cycle of length at most k ?

Acknowledgement. Research of the authors was supported by the European Research Council (ERC) grants “Rigorous Theory of Preprocessing,” reference 267959, and “PARAMTIGHT: Parameterized complexity and the search for tight complexity results,” reference 280152. We are also grateful to Saket Saurabh for helpful comments on this manuscript.

References

1. Ajtai, M.: The shortest vector problem in ℓ_2 is NP-hard for randomized reductions. In: STOC, pp. 10–19 (1998)
2. Alon, N., Yuster, R., Zwick, U.: Color-coding. *J. Assoc. Comput. Mach.* 42(4), 844–856 (1995)
3. Bodlaender, H.L., Downey, R.G., Fellows, M.R., Hallett, M.T., Wareham, H.T.: Parameterized complexity analysis in computational biology. *Computer Applications in the Biosciences* 11(1), 49–57 (1995)
4. Bodlaender, H.L., Downey, R.G., Fellows, M.R., Wareham, H.T.: The parameterized complexity of sequence alignment and consensus. *Theor. Comput. Sci.* 147(1, 2), 31–54 (1995)
5. Chen, J., Liu, Y., Lu, S.: An improved parameterized algorithm for the minimum node multiway cut problem. *Algorithmica* 55(1), 1–13 (2009)
6. Chen, J., Liu, Y., Lu, S., O’Sullivan, B., Razgon, I.: A fixed-parameter algorithm for the directed feedback vertex set problem. *J. ACM* 55(5), Art. 21, 19 (2008)
7. Chen, J., Liu, Y., Lu, S., O’Sullivan, B., Razgon, I.: A fixed-parameter algorithm for the directed feedback vertex set problem. In: STOC, pp. 177–186 (2008)
8. Cygan, M., Pilipczuk, M., Pilipczuk, M., Wojtaszczyk, J.O.: On Multiway Cut Parameterized above Lower Bounds. In: Marx, D., Rossmanith, P. (eds.) IPEC 2011. LNCS, vol. 7112, pp. 1–12. Springer, Heidelberg (2012)
9. Dahlhaus, E., Johnson, D.S., Papadimitriou, C.H., Seymour, P.D., Yannakakis, M.: The complexity of multiterminal cuts. *SIAM J. Comput.* 23(4), 864–894 (1994)
10. Dehne, F., Fellows, M., Langston, M.A., Rosamond, F., Stevens, K.: An $O(2^{O(k)}n^3)$ FPT Algorithm for the Undirected Feedback Vertex Set Problem. In: Wang, L. (ed.) COCOON 2005. LNCS, vol. 3595, pp. 859–869. Springer, Heidelberg (2005)
11. Downey, R.G., Fellows, M.R.: Fixed-parameter tractability and completeness I: Basic results. *SIAM J. Comput.* 24(4), 873–921 (1995)
12. Downey, R.G., Fellows, M.R.: Fixed-parameter tractability and completeness II: On completeness for $W[1]$. *Theor. Comput. Sci.* 141(1, 2), 109–131 (1995)

13. Downey, R.G., Fellows, M.R.: Parameterized computational feasibility. In: Feasible Mathematics II, pp. 219–244. Birkhäuser, Boston (1995)
14. Downey, R.G., Fellows, M.R.: Parameterized complexity. Springer, New York (1999)
15. El-Zahar, M.H., Schmerl, J.H.: On the size of jump-critical ordered sets. *Order* 1(1), 3–5 (1984)
16. Fellows, M.R.: Parameterized Complexity: The Main Ideas and Some Research Frontiers. In: Eades, P., Takaoka, T. (eds.) ISAAC 2001. LNCS, vol. 2223, pp. 291–307. Springer, Heidelberg (2001)
17. Fellows, M.R., Koblitz, N.: Fixed-Parameter Complexity and Cryptography. In: Moreno, O., Cohen, G., Mora, T. (eds.) AAECC 1993. LNCS, vol. 673, pp. 121–131. Springer, Heidelberg (1993)
18. Flum, J., Grohe, M.: Parameterized Complexity Theory. Texts in Theoretical Computer Science. An EATCS Series. Springer, Berlin (2006)
19. Fortune, S., Hopcroft, J., Wyllie, J.: The directed subgraph homeomorphism problem. *Theoret. Comput. Sci.* 10(2), 111–121 (1980)
20. Frick, M., Grohe, M.: Deciding first-order properties of locally tree-decomposable structures. *J. ACM* 48(6), 1184–1206 (2001)
21. Grohe, M.: Computing crossing numbers in quadratic time. In: STOC, pp. 231–236 (2001)
22. Grohe, M.: Computing crossing numbers in quadratic time. *J. Comput. Syst. Sci.* 68(2), 285–302 (2004)
23. Grohe, M., Kawarabayashi, K., Marx, D., Wollan, P.: Finding topological subgraphs is fixed-parameter tractable. In: Proceedings of the 43rd ACM Symposium on Theory of Computing, pp. 479–488 (2011)
24. Guillemot, S.: FPT algorithms for path-transversal and cycle-transversal problems. *Discrete Optimization* 8(1), 61–71 (2011)
25. Guo, J., Gramm, J., Hüffner, F., Niedermeier, R., Wernicke, S.: Compression-based fixed-parameter algorithms for feedback vertex set and edge bipartization. *J. Comput. Syst. Sci.* 72(8), 1386–1396 (2006)
26. Gustedt, J.: Well quasi ordering finite posets and formal languages. *J. Comb. Theory, Ser. B* 65(1), 111–124 (1995)
27. Hartvigsen, D.: The planar multiterminal cut problem. *Discrete Applied Mathematics* 85(3), 203–222 (1998)
28. Impagliazzo, R., Paturi, R., Zane, F.: Which problems have strongly exponential complexity? *Journal of Computer and System Sciences* 63(4), 512–530 (2001)
29. Johnson, D.S.: A catalog of complexity classes. In: Handbook of Theoretical Computer Science. in Algorithms and Complexity, vol. (A), pp. 67–161 (1990)
30. Kawarabayashi, K., Reed, B.A.: Computing crossing number in linear time. In: Proceedings of the 39th Annual ACM Symposium on Theory of Computing (STOC 2007), pp. 382–390. ACM (2007)
31. Klein, P.N., Marx, D.: Solving planar k -terminal cut in $O(n^{c\sqrt{k}})$ time. To appear in ICALP 2012 (2012)
32. Langston, M.A., Plaut, B.C.: On algorithmic applications of the immersion order: An overview of ongoing work presented at the Third Slovenian International Conference on Graph Theory. *Discrete Mathematics* 182(1-3), 191–196 (1998)
33. Lokshtanov, D., Marx, D.: Clustering with Local Restrictions. In: Aceto, L., Henzinger, M., Sgall, J. (eds.) ICALP 2011, Part I. LNCS, vol. 6755, pp. 785–797. Springer, Heidelberg (2011)

34. Marx, D.: A tight lower bound for planar multiway cut with fixed number of terminals. To appear in ICALP 2012 (2012)
35. Marx, D.: Parameterized graph separation problems. *Theoret. Comput. Sci.* 351(3), 394–406 (2006)
36. McCartin, C.: An improved algorithm for the jump number problem. *Inf. Process. Lett.* 79(2), 87–92 (2001)
37. Niedermeier, R.: Invitation to fixed-parameter algorithms. Oxford Lecture Series in Mathematics and its Applications, vol. 31. Oxford University Press, Oxford (2006)
38. Pietrzak, K.: On the parameterized complexity of the fixed alphabet shortest common supersequence and longest common subsequence problems. *J. Comput. Syst. Sci.* 67(4), 757–771 (2003)
39. Robertson, N., Seymour, P.D.: Graph minors. XIII. The disjoint paths problem. *J. Combin. Theory Ser. B* 63(1), 65–110 (1995)
40. Robertson, N., Seymour, P.D.: Graph minors XXIII. Nash-Williams’ immersion conjecture. *J. Comb. Theory, Ser. B* 100(2), 181–205 (2010)
41. Schrijver, A.: Finding k disjoint paths in a directed planar graph. *SIAM J. Comput.* 23(4), 780–788 (1994)
42. Scott, A.: On the parameterized complexity of finding short winning strategies in combinatorial games. Ph.D. thesis, University of Victoria (2009)
43. Thomassé, S.: A quadratic kernel for feedback vertex set. *ACM Trans. Algorithms* 6(2) (2010)
44. Vardy, A.: Algorithmic complexity in coding theory and the minimum distance problem. In: *STOC*, pp. 92–109 (1997)
45. Vertigan, D., Whittle, G.: Recognizing polymatroids associated with hypergraphs. *Combinatorics, Probability & Computing* 2, 519–530 (1993)

What's Next? Future Directions in Parameterized Complexity

Dániel Marx*

Computer and Automation Research Institute
Hungarian Academy of Sciences (MTA SZTAKI)
Budapest, Hungary
dmarx@cs.bme.hu

Abstract. The progress in parameterized complexity has been very significant in recent years, with new research questions and directions, such as kernelization lower bounds, appearing and receiving considerable attention. This speculative article tries to identify new directions that might become similar hot topics in the future. First, we point out that the search for optimality in parameterized complexity already has good foundations, but lots of interesting work can be still done in this area. The systematic study of kernelization became a very successful research direction in recent years. We look at what general conclusions one can draw from these results and we argue that the systematic study of other algorithmic techniques should be modeled after the study of kernelization. In particular, we set up a framework for understanding which problems can be solved by branching algorithms. Finally, we discuss that the domain of directed graph problems is a challenging area which can potentially see significant progress in the following years.

1 Introduction

*There was a guy whose name was Mike.
Loved math, surf, wine, and the like.
Once he climbed up a graph,
Took a photograph
And said: what a wonderful hike!*

Zsuzsa Mártonffy

The field of parameterized complexity progressed enormously since the publication of Downey and Fellows' monograph [44] in 1999. New techniques and new discoveries opened up new research directions and changed the field, sometimes in unexpected ways. Kernelization, a basic algorithmic technique for obtaining fixed-parameter tractability results, has evolved into a subfield of its own by better understanding of its applicability and the possibility of proving strong

* Dedicated to Michael R. Fellows on the occasion of his 60th birthday. Research supported by the European Research Council (ERC) grant "PARAMTIGHT: Parameterized complexity and the search for tight complexity results," reference 280152.

upper and lower bounds. As explained by Langston elsewhere in this volume [73], the fact that the Graph Minors Theory of Robertson and Seymour (culminating in papers [94, 93]) implies the existence of polynomial-time algorithms in a nonconstructive way was one of the early motivations for parameterized complexity. In the past decade, an entirely different aspect of Graph Minors Theory has been developed, which allows us for example to generalize in many cases fixed-parameter tractability results from planar graphs to H -minor free graphs (see the survey of Thilikos in this volume [96]). A useful product of this development is the concept of bidimensionality, which changed substantially the way we look at planar graph problems [36–38]. Even simple techniques can move the field into new directions: iterative compression, introduced by Reed et al. [92], turned out to be a key step in proving the fixed-parameter tractability of important problems such as BIPARTITE DELETION [92], ALMOST 2SAT [91], and DIRECTED FEEDBACK VERTEX SET [25], and changed the way we look at problems involving deletions.

One could list several other aspects in which the field evolved and changed in the past decade. However, the purpose of this article is *not* to review these developments. Rather than that, the purpose of this article is to propose some new directions for future research. Only time and further work can tell if these directions are as fruitful as the ones listed above.

The first topic we discuss is the optimality program of parameterized complexity: understanding quantitatively what is the best we can achieve for a particular problem. That is, instead of just establishing fixed-parameter tractability, we eventually want to understand the best possible $f(k)$ in the running time. This is not really a new direction, as the literature contains several results of this type. However, we feel that it is important to emphasize here that the search for optimality is a viable and very timely research program which should be guiding the development of the field in the following years.

Kernelization is perhaps the most practical technique in the arsenal of fixed-parameter tractability, thus it is not surprising that its methods and applicability have received particular attention in the literature. In the past few years, research on kernelization has increased enormously after it had been realized that the existence of polynomial kernels is a mathematically deep and very fruitful research question, both from the algorithmic and complexity points of view. A detailed overview of the results on kernelization is beyond the scope of this article; the reader is referred to [84, 12, 76] for a survey of recent results. However, we briefly review kernelization from the point of view of the optimality program. What we would like to point out is that the study of kernelization should be interpreted as a search for a tight understanding of the power of kernelization. That is, the question guiding our research is not which problems *can* be solved by kernelization, but rather which problems *should* be solved by kernelization.

Kernelization is just one technique in parameterized complexity and its systematic study opened up a whole new world of research questions. Could it be that exploring other basic techniques turns out to be as fruitful as the study of kernelization? Besides kernelization, branching is the most often used technique,

thus it could be the next natural target for rigorous analysis. We propose a framework in which one can study whether a problem can be solved by branching or not. Based on what we have learned from the study of kernelization, one should look at the study of branching also from the viewpoint of optimality: the goal is to understand for which problems is branching the right way of solution. The description and discussion of this framework is the only part of the paper containing new technical ideas. The presentation of this framework is intentionally kept somewhat informal, as going into the details of irrelevant technical issues would distract from the main message.

The last direction we discuss is the study of algorithmic problems on directed graphs. Perhaps it is premature to call such a wide area with disconnected results as a research direction. However, we would like to point out the enormous potential in pursuing questions in this direction. Problems on directed graphs are much more challenging than their undirected counterparts, as we are in a completely different world where many of the usual tools do not help at all. Still, there are directed problems that have been tackled successfully in recent years, for example, DIRECTED FEEDBACK VERTEX SET [25] or DIRECTED MULTIWAY CUT [26]. This suggests that it is not hopeless to expect further progress on directed graphs, or even a general theory that is applicable for several problems.

2 The Optimality Program

Recall that a parameterized problem is *fixed-parameter tractable (FPT)* with a given parameterization if there is an algorithm with running time $f(k) \cdot n^{O(1)}$, where n is the size of the instance, k is the value of the parameter associated with the instance, and f is an arbitrary computable function depending only on the parameter k (see the monographs [44, 50, 87] or the survey [41] in this volume for more background). That is, the problem can be solved in polynomial time for every fixed value of the parameter k and the exponent *does not* depend on the parameter. Intuitively, we would like fixed-parameter tractability to express that the problem has an “efficient” or “practical” algorithm for small values of k . However, the definition only requires that f is computable and it can be any fast growing, ugly function. And this is not only a hypothetical possibility: the early motivation for parameterized complexity came from algorithmic results based on the Graph Minors Theory of Robertson and Seymour, and the $f(k)$ in these algorithms are typically astronomical towers of exponentials, far beyond any hope of practical use.

For most FPT problems, however, there are algorithms with “well-behaving” $f(k)$. In many cases, $f(k)$ is c^k for some reasonably small constant $c > 0$. For example, VERTEX COVER can be solved in time $1.2738^k \cdot n^{O(1)}$ [23]. Sometimes the function $f(k)$ is even subexponential, e.g., $c^{\sqrt{k}}$. It happens very often that by understanding a problem better or by using more advanced techniques, better and better FPT algorithms are developed for the same problem and a kind of “race” is established to make $f(k)$ as small as possible. Clearly, it would be very useful to know if the current best algorithm can be improved further or it has

already hit some fundamental barrier. If a problem is NP-hard, then we cannot expect $f(k)$ to be polynomial. But is it possible that something very close to polynomial, say $k^{\log \log \log k}$ can be reached? Or are there problems for which the best possible $f(k)$ is very bad, say, of the form $2^{2^{\Omega(k)}}$? The optimality program tries to understand and answer such questions.

In recent years, a lower bound technology was developed, which, in many cases, is able to demonstrate the optimality of fixed-parameter tractability results. We sketch how such lower bounds can be proved using a complexity-theoretic assumption called Exponential Time Hypothesis (ETH). (An alternate way to discuss these results is via the complexity class $M[1]$ and for some of the results even the weaker $FPT \neq W[1]$ hypothesis is sufficient. However, to keep our discussion focused, we describe only results based on ETH here.) For our purposes, ETH can be stated as follows:

Conjecture 2.1 (Exponential Time Hypothesis [63]). *3-SAT cannot be solved in time $2^{o(m)}$, where m is the number of clauses.*

This conjecture was first formulated by Impagliazzo, Paturi, and Zane [63]. More precisely, they stated a version of the conjecture saying that there is no $2^{o(n)} \cdot m^{O(1)}$ time algorithm, where n is the number of variables, and showed by a reduction called the Sparsification Lemma that the two versions of the conjecture are equivalent. Although there is no universal consensus in accepting ETH (compared to more established conjectures such as $P \neq NP$), it is consistent with our current knowledge: after several rounds of improvement, the best algorithm for n -variable m -clause 3-SAT has running time $O(1.30704^n)$ [60] and no algorithm with subexponential running time in m seems to be in sight.

If we accept ETH, then we can obtain lower bounds for other problems through reductions. Let us observe that standard NP-hardness reductions from 3-SAT to, say, INDEPENDENT SET are sensitive to the number of clauses in the input instance. That is, there is a polynomial-time algorithm that, given an m -clause 3SAT instance ϕ , constructs a $O(m)$ -vertex graph G and an integer k such that ϕ is satisfiable if and only if G has an independent set of size k . Therefore, assuming ETH, INDEPENDENT SET cannot be solved in time $2^{o(n)}$ on n -vertex graphs, as this algorithm together with the reduction from 3SAT would give a $2^{o(m)}$ time algorithm for m -clause 3SAT. If we look at the literature on NP-hardness proofs, then we can see that many other hardness proofs have this property. From these hardness proofs, we can obtain results such as the following:

Corollary 2.2. *Assuming ETH, there is no $2^{o(n)}$ time algorithm for INDEPENDENT SET, CLIQUE, DOMINATING SET, HAMILTONIAN PATH on n -vertex graphs.*

This means that every algorithm for these problems has to run in time exponential in the number of vertices or, in other words, there are no subexponential FPT algorithms parameterized by the number of the vertices. A colloquial term for algorithms that solve the problem in exponential time in the number of vertices, possibly in a smart way, is “exact exponential-time algorithms” [52];

Corollary 2.2 can be interpreted as a lower bound on exact algorithms. However, as the number of vertices is an upper bound on the size of the solution, it also follows that there are no subexponential FPT algorithms parameterized by the size of the solution:

Corollary 2.3. *Assuming ETH, there is no $2^{o(k)} \cdot n^{O(1)}$ time algorithm for INDEPENDENT SET, CLIQUE, DOMINATING SET, and k -PATH, where k is the size of the solution to be found.*

There are FPT problems for which there are subexponential-time parameterized algorithms. This is very common for planar problems: all the problems in Corollary 2.3 are known to be solvable in time $2^{O(\sqrt{k})} \cdot n^{O(1)}$ on planar graphs.¹ There are two main approaches for obtaining running time of this form on planar graphs: using planar separator results [3] and bidimensionality theory [36]. On the complexity side, if we look at the proofs showing the NP-hardness of these problems in planar graphs, then all of them involve “crossover gadgets” to deal with planarity. These gadgets induce a blowup in the size of the constructed instance: it is no longer linear in the number of clauses, but quadratic. Therefore, we get weaker lower bounds: we can only rule out the existence of algorithms with running time $2^{O(\sqrt{k})} \cdot n^{O(1)}$.

Corollary 2.4. *Assuming ETH, there is no $2^{o(\sqrt{k})} \cdot n^{O(1)}$ time algorithm for INDEPENDENT SET, DOMINATING SET, and k -PATH on planar graphs, where k is the size of the solution to be found.*

Note that these lower bounds match the known $2^{O(\sqrt{k})} \cdot n^{O(1)}$ time algorithms (up to the constant hidden by the big-O notation). These matching bounds can be considered a major success of the optimality program so far. Initially looking at planar problems, it is not obvious why square root is the function that should appear in the running time, but we have learned that this is an inherent feature of planarity and now we have a good understanding of both the upper bounds and the lower bounds.

A planar problem which is not fully understood yet is SUBGRAPH ISOMORPHISM: given graphs H and G , does G has a subgraph isomorphic to H ? On planar graphs, the problem is known to be solvable in time $2^{O(k)} \cdot n^{O(1)}$ [39], where k is the number of vertices of H (improving an earlier $k^{O(k)} \cdot n^{O(1)}$ algorithm [45]). Could it be that square root appears in this problem as well and the running time can be improved further to $2^{O(\sqrt{k})} \cdot n^{O(1)}$? There is no known complexity result ruling out this possibility. Furthermore, significantly new techniques would be required to rule out the existence of a $2^{o(k)} \cdot n^{O(1)}$ algorithm for the problem: as the problem is planar, typical reductions need to introduce crossover gadgets, which would create a blowup in the size of the instance.

Subexponential-time FPT results are fairly standard for planar problems. It is much more surprising if a problem on general graphs admits a subexponential-time algorithm. Very recently, this turned out to be the case for the CHORDAL

¹ Actually, CLIQUE can be solved in polynomial time on planar graphs.

COMPLETION problem (given a graph G and an integer k , decide if G can be made chordal by adding at most k edges). Various $2^{O(k)} \cdot n^{O(1)}$ time algorithms are known for the problem [17, 65, 15]. Fomin and Villanger [54] gave a significant improvement by presenting a $2^{O(\sqrt{k} \log k)} \cdot n^{O(1)}$ time algorithm. It is an interesting question whether this running time can be further improved. As observed in [54], the NP-hardness proofs imply that, assuming ETH, there is no $2^{o(k^{1/6})} \cdot n^{O(1)}$ time algorithm. Therefore, currently there is a large gap between the best upper and lower bounds.

Obtaining lower bounds of the form $2^{o(k)} n^{O(1)}$ or $2^{o(\sqrt{k})} n^{O(1)}$ on parameterized problems generally follows from the known NP-hardness reductions. However, there are some parameterized problems where $f(k)$ is “slightly superexponential” in the best known running time: $f(k)$ is of the form $k^{O(k)} = 2^{O(k \log k)}$. Algorithms with this running time naturally occur when a search tree of height at most k and branching factor at most k is explored, or when all possible permutations, partitions, or matchings of a k element set are enumerated. In many cases, the $f(k)$ running time was later improved to $2^{O(k)}$, often with significant extra work or with the introduction of a new technique. We have seen an example of this with the SUBGRAPH ISOMORPHISM problem on planar graphs. Another example: Monien [85] in 1985 gave a $k! \cdot n^{O(1)}$ time algorithm for finding a cycle of length k in a graph on n vertices. Alon, Yuster, and Zwick [6] introduced the color coding technique in 1995 and used it to show that a cycle of length k can be found in time $2^{O(k)} \cdot n^{O(1)}$. A very recent example is the case of the HAMILTONIAN CYCLE problem parameterized by treewidth. A $w^{O(w)} \cdot n^{O(1)}$ time algorithm for graphs of treewidth w follows from standard dynamic programming techniques (see e.g., [50]). Very recently, Cygan et al. [30] introduced an elegant new technique called cut and count, and used it to design a (randomized) algorithm that, given a tree decomposition of width w , solves the problem in time $4^w \cdot n^{O(1)}$.

However, there are still a number of problems where the best running time seems to be “stuck” at $2^{O(k \log k)} \cdot n^{O(1)}$. Recently, for some of these problems matching lower bounds excluding running times of the form $2^{o(k \log k)} \cdot n^{O(1)}$ were obtained under ETH [75] (see also [30] for further examples), showing the optimality of these algorithms.

- The pattern matching problem CLOSEST STRING (given k strings over an alphabet Σ and an integer d , decide if there is a string whose Hamming-distance is at most d from each of the k strings) is known to be solvable in time $2^{O(d \log d)} \cdot n^{O(1)}$ [57] or $2^{O(d \log |\Sigma|)} \cdot n^{O(1)}$ [77]. Assuming ETH, there is no $2^{o(d \log d)} \cdot n^{O(1)}$ and $2^{o(d \log |\Sigma|)} \cdot n^{O(1)}$ time algorithms [75].
- The graph embedding problem DISTORTION (decide whether a graph G has a metric embedding into the integers with distortion at most d) can be solved in time $2^{O(d \log d)} \cdot n^{O(1)}$ [47]. Assuming ETH, there is no $2^{o(d \log d)} \cdot n^{O(1)}$ time algorithm [75].
- The DISJOINT PATHS problem can be solved in time in time $2^{O(w \log w)} \cdot n^{O(1)}$ on graphs of treewidth at most w [95]. Assuming ETH, there is no $2^{o(w \log w)} \cdot n^{O(1)}$ time algorithm [75].

We expect that many further results of this form can be obtained by using the framework of [75]. Thus the existence of parameterized problems requiring “slightly superexponential” time $2^{O(k \log k)} \cdot |I|^{O(1)}$ is not a shortcoming of algorithm design or a pathological situation, but an unavoidable feature of the landscape of parameterized complexity.

The results discussed so far show the optimality of some $2^{O(k)} \cdot n^{O(1)}$, $2^{O(\sqrt{k})} \cdot n^{O(1)}$, and $2^{O(k \log k)} \cdot n^{O(1)}$ time algorithms. Are there natural problems for which the optimum running time is of some other form, say, $2^{O(k^2)} \cdot n^{O(1)}$ or $2^{2^{O(k)}} \cdot n^{O(1)}$? The curious problem CLIQUE-OR-INDEPENDENT-SET (given a graph G and an integer k , is there a set of k vertices that induce a clique *or* an independent set?) can be solved in time $2^{O(k^2)} \cdot n^{O(1)}$ using a simple Ramsey argument ([69, 67]), thus it could be a candidate problem where this form of running time is optimal. PLANAR DELETION (delete k vertices to make the graph planar) could be a candidate for a natural problem where double-exponential dependence on k is necessary. The fixed-parameter tractability results for PLANAR DELETION [83, 66] depend on solving the problem on bounded-treewidth graphs, and it seems that the natural algorithm based on destroying all K_5 and $K_{3,3}$ subdivisions have double-exponential dependence on treewidth.

A more ambitious project is to understand the exact constants in the function $f(k)$ for the problem: for example, what is the smallest $c > 0$ such that there is a $c^k \cdot n^{O(1)}$ time algorithm for the problem? Let us note first that obtaining such results is very different and much more challenging than proving lower bounds of the form, say, $2^{o(k)} \cdot n^{O(1)}$. The problem is that determining the best possible c is machine-model dependent in the sense that it is not robust under polynomial-transformations of the running time. That is, a $4^k \cdot n^{O(1)}$ running time is just the square of $2^k \cdot n^{O(1)}$. ETH as formulated in Conjecture 2.1, however, is invariant under polynomial transformations of the running time: any polynomial of $2^{o(m)}$ is still $2^{o(m)}$. Therefore, it seems unlikely that such a coarse conjecture would give an easy way of proving the fine distinctions between running times $c^k \cdot n^{O(1)}$ for different values of c . A more suitable conjecture is the Strong Exponential Time Hypothesis (SETH); for the purposes of this paper, we can state it the following way:

Conjecture 2.5 (Strong Exponential Time Hypothesis [63, 18]). *There is no $(2 - \epsilon)^n \cdot m^{O(1)}$ time algorithm for n -variable m -clause SAT for any $\epsilon > 0$.*

Note that here SAT is the satisfiability problem with unbounded clause size. For fixed clause size, there are better algorithms, see e.g., [60]. Lokshtanov et al. [74] used SETH to prove tight lower bounds on algorithms working on tree decompositions. Suppose that we want to solve a problem on a graph G and a tree decomposition of width w of G is given in the input. Assuming SETH, for every $\epsilon > 0$

- INDEPENDENT SET cannot be solved in $(2 - \epsilon)^w |V(G)|^{O(1)}$ time,
- DOMINATING SET cannot be solved in $(3 - \epsilon)^w |V(G)|^{O(1)}$ time,
- MAX CUT cannot be solved in $(2 - \epsilon)^w |V(G)|^{O(1)}$ time,
- ODD CYCLE TRANSVERSAL cannot be solved in $(3 - \epsilon)^w |V(G)|^{O(1)}$ time,

- For any $q \geq 3$, q -COLORING cannot be solved in $(q - \epsilon)^w |V(G)|^{O(1)}$ time,
- PARTITION INTO TRIANGLES cannot be solved in $(2 - \epsilon)^w |V(G)|^{O(1)}$ time.

These lower bounds match the best known algorithms for the problem (up to the ϵ in the base of the exponent). Some further lower bounds of this form can be found in [30]. It seems to be a very different and significantly more challenging task to prove such tight results for problems parameterized by the size of the solution (instead of treewidth). The natural targets for such lower bounds are problems where the best known algorithms have running times of the form $c^k \cdot n^{O(1)}$ for some integer c . Cygan et al. [30] gave such (randomized) algorithms for a number of problems using the technique of cut and count.

The optimality results we have discussed so far make fixed-parameter tractability quantitative: we not only know now that the problem is FPT, but we also know what the best $f(k)$ in the running time can be. Another aspect of the optimality program is to make W[1]-hardness results quantitative. That is, instead of just knowing that the problem is not FPT and therefore the parameter has to appear in the exponent of the running time, we would like to know how exactly the exponent should depend on the parameter. A W[1]-hardness result by itself does not rule out the possibility that the problem can be solved in, say, time $2^k \cdot n^{O(\log \log \log \log k)}$, which would be “morally equivalent” to fixed-parameter tractability.

The Exponential Time Hypothesis can be used to give a tight lower bound on the exponent of the running time. Chen et al. [22] showed that for the CLIQUE problem the $n^{O(k)}$ brute force algorithm is already optimal in this respect:

Theorem 2.6. *Assuming ETH, CLIQUE cannot be solved in time $f(k) \cdot n^{o(k)}$ for any computable function f .*

Using parameterized reductions, we can transfer the lower bound of Theorem 2.6 to other problems. The exact form of the lower bound depends on how the parameterized reduction changes the parameter. For the following problems, the reductions increase the parameter at most by a constant factor, thus we get a lower bound of the same form:

Theorem 2.7. *Assuming ETH, INDEPENDENT SET and DOMINATING SET cannot be solved in time $f(k) \cdot n^{o(k)}$ for any computable function f .*

On the other hand, if the reduction increases the parameter by more than a constant factor, then the lower bound gets weaker. For example, a reduction from CLIQUE (on general graphs) to DOMINATING SET on unit disk graphs was presented in [79], which increases the parameter from k to $O(k^2)$. Therefore, we have the following lower bound:

Theorem 2.8. *Assuming ETH, DOMINATING SET on unit disk graphs cannot be solved in time $f(k) \cdot n^{o(\sqrt{k})}$ for any computable function f .*

As DOMINATING SET on unit disk graphs can be solved in time $n^{O(\sqrt{k})}$ [4], Theorem 2.8 is tight. Thus, similarly to many planar problems, the appearance

of the square root in the running time can be an inherent feature of geometric problems.

Most W[1]-hardness results in the literature are from CLIQUE (or INDEPENDENT SET, which is the same). Therefore, by analyzing how the parameter changes in the reduction, we can extract lower bounds similar to the ones above by transferring Theorem 2.6 to the problem at hand. One should examine it separately for each problem whether the lower bound obtained this way is tight or not. Many of the more involved reductions from CLIQUE use edge selection gadgets (see e.g., [48, 51, 79]). As a clique of size k has $\Theta(k^2)$ edges, this means that the reduction typically increases the parameter to $\Theta(k^2)$ at least and, similarly to Theorem 2.8, what we can conclude is that there is no $f(k)n^{o(\sqrt{k})}$ time algorithm for the target problem (unless ETH fails). If we want to obtain stronger bounds on the exponent, then we have to avoid the quadratic blow up of the parameter and do the reduction from a different problem. Many of the reductions from CLIQUE can be turned into a reduction from the more general SUBGRAPH ISOMORPHISM (Given two graphs H and G , decide if H is a subgraph of G). In a reduction from SUBGRAPH ISOMORPHISM, we need $|E(H)|$ edge selection gadgets, which usually implies that the new parameter is $\Theta(|E(H)|)$. Thus the following lower bound on SUBGRAPH ISOMORPHISM, parameterized by the number of edges in H , could be used to obtain tighter lower bounds compared to those coming from the reduction from CLIQUE.

Theorem 2.9 ([81]). *If SUBGRAPH ISOMORPHISM can be solved in time $f(k)n^{o(k/\log k)}$, where f is an arbitrary function and k is the number of edges of the smaller graph H , then ETH fails.*

We remark that it is an interesting open question if the factor $\log k$ in the exponent can be removed, making this result tight (and also making the results following from Theorem 2.9 tighter).

CLOSEST SUBSTRING (a generalization of CLOSEST STRING) is an extreme example where reductions increase the parameter exponentially or even double exponentially, and therefore we obtain very weak lower bounds. In this problem, the input consists of strings s_1, \dots, s_t over an alphabet Σ and integers L and d . The task is to find a string s of length L such that every s_i has a consecutive substring s'_i of length L with Hamming-distance at most d from s .

Let us restrict our attention to the case where the alphabet is of constant size, say binary. Marx [80] gave a reduction from CLIQUE to CLOSEST SUBSTRING where $d = 2^{O(k)}$ and $t = 2^{2^{O(k)}}$ in the constructed instance (k is the size of the clique we are looking for in the original instance). Therefore, we get weak lower bounds with only $o(\log d)$ and $o(\log \log k)$ in the exponent. Surprisingly, these lower bounds are actually tight, as there are algorithms matching these bounds.

Theorem 2.10 ([80]). *CLOSEST SUBSTRING over an alphabet of constant size can be solved in time $f(d)n^{O(\log d)}$ or in $f(d, k)n^{O(\log \log k)}$. Furthermore, assuming ETH, there are no algorithms for the problem with running time $f(k, d)n^{o(\log d)}$ or $f(k, d)n^{o(\log \log k)}$.*

While the results in Theorems 2.6 and 2.8 are asymptotically tight, they do not tell us the exact form of the exponent, that is, we do not know what the smallest c is such that the problems can be solved in time n^{ck} . However, assuming SETH, stronger bounds of this form can be obtained. Specifically, Pătraşcu and Williams [88] obtained the following bound for DOMINATING SET under SETH.

Theorem 2.11 ([88]). *Assuming SETH, there is no $O(n^{k-\epsilon})$ time algorithm for DOMINATING SET for any $\epsilon > 0$ and $k \geq 2$.*

3 Kernelization from the Viewpoint of Optimality

Kernelization is one of the most basic and most practical algorithmic techniques in parameterized complexity. Recall that a *kernelization* for a parameterized problem P is a polynomial-time algorithm that, given an instance I of P with parameter k , produces another instance I' of P with parameter k' such that (1) I is a yes-instance if and only if I' is a yes-instance, (2) the size of I' is at most $f(k)$ for some computable function f , and (3) k' is at most $f(k)$. Intuitively, one can think of a kernelization as a fast preprocessing algorithm producing a small “hard core” of the problem that needs to be solved. We say that a kernelization is an $f(k)$ -kernel if the size of I' is at most $f(k)$. For graph problems, we also use the term $f(k)$ -vertex-kernel to indicate that I' has at most $f(k)$ vertices.

If a parameterized problem admits a kernel, then this immediately implies that the problem is FPT. We can use the kernelization algorithm to produce an equivalent instance I' of size at most $f(k)$, and then we can use any brute force algorithm to solve I' in time that can be bounded by a function of k (assuming the problem is decidable). More surprisingly, a folklore result shows that the reverse direction is also true:

Theorem 3.1. *A decidable parameterized problem has a kernel if and only if it is FPT.*

Proof. We have seen the forward direction above. For the reverse direction, suppose that a parameterized problem can be solved in time $f(k) \cdot n^c$ for some computable function $f(k)$ and constant c . Given an instance I of the problem, let us simulate this algorithm for n^{c+1} steps. If the algorithm terminates during this simulation, then we can produce a kernel by outputting a trivial yes- or a trivial no-instance. If the $f(k) \cdot n^c$ time algorithm does not terminate in n^{c+1} steps, then $n < f(k)$. This means that I itself is a kernel with size at most $f(k)$. \square

What does Theorem 3.1 tell us? It suggests that every FPT result can be explained as a kernelization together with an exact algorithm. Thus the study of fixed-parameter tractability can be reduced to the study of kernelization algorithms and exact exponential-time algorithms (or in other words, parameterization by the size of the instance). Given the breadth of techniques in parameterized complexity that does not seem to have anything to do with these two

concepts (e.g., color coding, iterative compression, and algebraic techniques), this is a somewhat disheartening and suspicious claim.

Let us revisit this claim from the viewpoint of the optimality program. It is true that every fixed-parameter tractability result can be obtained as a combination of kernelization and exact algorithms, but is it the *right way* of solving the problem? That is, can we get the best possible (or at least a reasonable good) $f(k)$ in the running time this way? For some problems this seems to be the case. For example, a classical result of Nemhauser and Trotter [86] shows that VERTEX COVER admits a $2k$ -vertex-kernel and can be solved trivially in time $2^{O(n)}$ on n -vertex graphs. This results in a $2^{O(k)} \cdot n^{O(1)}$ time algorithm, which is the optimal form of the running time by Corollary 2.3. For DOMINATING SET on planar graphs, several $O(k)$ -vertex-kernels are known and the problem can be solved in time $2^{O(\sqrt{n})}$ on n -vertex planar graphs either by treewidth techniques or by using planar separator theorems. This combination gives us $2^{O(\sqrt{k})} \cdot n^{O(1)}$ time algorithms, which matches the lower bound of Corollary 2.4.

For other problems, however, we cannot reach the best possible running time using this combination. In order to show this, we need a way of proving lower bounds on the size of kernels that can be achieved. Bodlaender et al. [13], using the work of Fortnow and Santhanam [55], developed a technique for showing (modulo a complexity assumption) that certain problems do not admit kernels of polynomial sizes. This result started a whole new line of research and the technique has been subsequently used in several papers to prove similar lower bounds. We state only one such result here:

Theorem 3.2 ([13]). *Assuming $\text{coNP} \not\subseteq \text{NP/poly}$, there is no $k^{O(1)}$ -kernel for k -PATH.*

The assumption $\text{coNP} \not\subseteq \text{NP/poly}$ is a fairly standard complexity assumption, for example, if it is false, then the polynomial hierarchy collapses [98].

The k -PATH problem is known to be solvable in time $2^{O(k)} \cdot n^{O(1)}$ by various techniques [6, 10, 97]. Can we match this running time by a combination of kernelization and exact algorithms? Clearly, we can solve k -Path by a brute force exact algorithm in time $n^{O(k)}$. By Theorem 3.2, we cannot produce a kernel with k^c vertices for any constant c , thus this combination cannot even guarantee a running time of $k^{ck} \cdot n^{O(1)}$ for any constant c . In other words, even though Theorem 3.1 shows that k -PATH has a kernelization algorithm and therefore in principle we could obtain FPT algorithms for the problem by kernelization followed by an exact algorithm, this is not the right combination of techniques to solve the problem, as it cannot reach the best possible running time $2^{O(k)} \cdot n^{O(1)}$.

We can argue similarly for other problems where the existence of a polynomial kernel can be ruled out by a result analogous to Theorem 3.2. But what about problems for which polynomial kernels do exist? Very recently, some results appeared that give tight lower bounds for problems admitting polynomial kernels. Recall that given a collection of sets of size d of a set of elements and an integer k , the d -SET COVER asks for a set of at most k elements that intersects every set in the input, while the d -SET PACKING problem asks for k pairwise-disjoint sets from the collection. For both problems, algorithms based on the Sunflower

Lemma give kernels containing at most k^d sets (see e.g., [34]). The following results show that this is essentially best possible:

Theorem 3.3 ([35]). *Assuming $\text{coNP} \not\subseteq \text{NP/poly}$, there is no $O(k^{d-\epsilon})$ -kernel for d -SET COVER for any $d \geq 3$ and $\epsilon > 0$.*

Theorem 3.4 ([34]). *Assuming $\text{coNP} \not\subseteq \text{NP/poly}$, there is no $O(k^{d-\epsilon})$ -kernel for d -SET PACKING for any $d \geq 3$ and $\epsilon > 0$.*

The d -SET COVER problem can be solved in time $d^k \cdot n^{O(1)}$ by simple branching and d -SET PACKING can be solved in time $2^{O(dk)} \cdot n^{O(1)}$ for example by color coding. Can we match these running times by a combination of kernelization and exact algorithms? It is not clear at this point. Both problems can be solved in time $2^{O(n)}$ if n is the number of elements (this is obvious for d -SET COVER, as we can try every subset of the elements; for d -SET PACKING, this follows from standard dynamic programming techniques). Therefore, the questions is whether kernels with $O(k)$ elements exist for these problems. Theorems 3.3–3.4 do not rule out this possibility, as they give a lower bound on the number sets only. Note that the current best upper bounds on the number of elements in the kernel are far from being $O(k)$ [1, 2]. It is a very interesting and challenging question for further research to understand what the best possible bound is in terms of the number of elements. From the viewpoint of the optimality program, one needs to answer this question in order to evaluate whether kernelization is the right way of solving these problems, or other techniques such as branching and color coding are inherently necessary to achieve the best possible running time.

Finally, for problems that admit linear (vertex-)kernels, one would like to know the best possible constant factor. For example, is there a $(2 - \epsilon)k$ -vertex-kernel for VERTEX COVER? There is a simple 2-approximation for this problem and there is no $(2 - \epsilon)$ -approximation under the Unique Games Conjecture [68]. It seems to be too much of a coincidence that the same number appears both in the best kernel and the best approximation. This could be the sign that there are some deep connections that we are unaware of at the moment.

Chen et al. [20] proposed an elegant argument for proving lower bounds on kernel size. The *parametric dual* of a parameterized problem with respect to a size function s is the same problem, but now we consider $s - k$ the parameter instead of k . For example, the parametric dual of VERTEX COVER with respect to the number of vertices is INDEPENDENT SET (since there is a vertex cover of size k in an n -vertex graph if and only if there is an independent set of size $n - k$). Chen et al. [20] showed that if a parameterized problem and its dual both admit small kernels of linear size, then one can solve the instance by repeated applications of the two kernelization algorithms. This technique is very useful for planar or bounded-degree problems, as for these classes it is fairly natural that both the problem and its parametric dual have linear kernels. Let us state as an example a few lower bounds that follow from this technique:

Theorem 3.5. [20] *Assuming $P \neq NP$, for any $\epsilon > 0$*

- VERTEX COVER on planar graphs does not have a $(\frac{4}{3} - \epsilon)k$ -vertex-kernel.
- VERTEX COVER on planar triangle-free graphs does not have a $(\frac{3}{2} - \epsilon)k$ -vertex-kernel.
- INDEPENDENT SET on planar graphs does not have a $(2 - \epsilon)k$ -vertex-kernel.
- DOMINATING SET on planar graphs does not have a $(2 - \epsilon)k$ -vertex-kernel.

Note, however, that these result do not give lower bounds on kernelization for general graphs: a kernelization algorithm for general graphs can transform a planar instance into a nonplanar one, hence it is not necessarily a correct kernelization algorithm for the planar problem as well.

We conclude this section by pointing out two technical issues that have arisen in the study of kernelization. In the definition of kernelization, we want to bound the size of the constructed instance. However, we might want to bound some other measure instead, for example, the number of vertices in the graph. From the practical point of view, for most graph-theoretical problems the time required for the exact solution of the kernel is mainly influenced by the number of vertices, thus it makes sense to focus on reducing the number of vertices. On the other hand, bounding the size of the instance seems to be mathematically more robust question, for example, the techniques of [13, 35] give primarily lower bounds on the size of the instance. Both kind of bounds are worth studying, but we have to make a clear distinction between the two types of results and realize the different consequences.

Another technical issue is the bound on the parameter in the kernel. Originally, Downey and Fellows [44] required that the parameter of the kernel is at most the parameter of the original instance. This makes sense: as we imagine that the parameter measures the hardness of the instance, we do not want the preprocessing to increase it. Later, e.g., in [14, 13] a more liberal definition is given, where we only require that the new parameter is bounded by a function of the old parameter (we used this definition in the beginning of the section). An advantage of this definition is that it is robust with respect to polynomial transformations of the kernel. For example, we can create a polynomial-size kernel that is an instance of some other problem (this is sometimes called a *bikernel*) and then use a polynomial-time reduction to transform it into an instance of the original problem. This results in a polynomial-size kernel, but the parameter can increase in the reduction. Allowing such arguments in proving the existence of polynomial-size kernels makes the theory more robust and mathematically more natural, although it weakens the connection with practical preprocessing. One has to be aware of this difference and interpret the results accordingly.

4 Branching Algorithms

Besides kernelization, the technique of “bounded-depth search trees” is perhaps the most basic method for showing that a problem is fixed-parameter tractable. Let us recall how this technique works in the case of VERTEX COVER. Let G

be a graph where a vertex cover of size k has to be found. Let $e = uv$ be an arbitrary edge of G . Clearly, every vertex cover contains either u or v (or both). Therefore, we branch into two directions. In the first branch, we assume that u is in the vertex cover, hence we try to find recursively a vertex cover of size $k - 1$ in $G \setminus u$. In the second branch, we assume that v is in the vertex cover and try to find recursively a vertex cover of size $k - 1$ in $G \setminus v$. Clearly, if there is a solution, at least one of the two branches finds a solution. We repeat this branching step until there is no edge in the graph or k becomes 0. Running this recursive process creates a search tree where each node has at most two children. The crucial property to observe is that the height of the search tree is at most k : the parameter strictly decreases in each step. Therefore, the search tree has at most 2^k leaves and hence $O(2^k)$ nodes. Each recursion step can be done in polynomial time, hence it follows that the total running time is $2^k \cdot n^{O(1)}$. The d -SET COVER problem is a generalization of VERTEX COVER: given sets of size d , we have to find k elements that hit every set. In a similar way, one can obtain a $d^k \cdot n^{O(1)}$ algorithm for d -SET COVER by selecting a set and branching on which element of the set is included in the solution.

In summary, the main idea of the bounded-depth search tree technique is to reduce the instance into a bounded number of instances with strictly smaller parameter values. If the reduction creates at most c instances, then the running time is $c^k \cdot n^{O(1)}$. In some cases, the number of directions we branch into depends also on the parameter. For example, if we create at most k instances in each step, then we can bound the running time by $k^k \cdot n^{O(1)}$. The $d^d \cdot n^{O(1)}$ time algorithm for CLOSEST STRING [57] mentioned in Section 2 is an example of such a branching algorithm.

Seeing how fruitful the systematic analysis of kernelization turned out to be, one wonders why there haven't been any systematic analysis of the applicability of branching algorithms. The purpose of this section is to propose a framework in which this question can be studied. What we have learned in the study of kernelization is that one should pay attention to optimality: the question is not whether branching algorithms *can* be used to solve a problem, but whether it is the *right way* of solving the problem. Therefore, here we stick to the study of $c^k \cdot n^{O(1)}$ time algorithms that branch into a constant number of directions. In particular, we are not interested in the question whether there is a $k^k \cdot n^{O(1)}$ time branching algorithm for a problem that can be solved in time $c^k \cdot n^{O(1)}$ by other techniques (because such a branching algorithm would be far from being the optimal way of solving the problem).

Let us formalize first the notion of a branching rule.

Definition 4.1. *Let (I, k) be an instance of a parameterized problem with $k > 1$. A c -way branching rule for some constant c is a polynomial-time algorithm that, given instance I , produces instances $(I_1, k_1), \dots, (I_c, k_c)$ such that*

1. $|I_i| \leq |I|$ for every $1 \leq i \leq c$,
2. $k_i < k$ for every $1 \leq i \leq c$,
3. (I, k) is a yes-instance if and only if (I_i, k_i) is a yes-instance for at least one $1 \leq i \leq c$.

It is easy to see that if a parameterized problem has a c -way branching rule, then we can solve the problem in time $c^k \cdot n^{O(1)}$ (assuming the problem is polynomial-time solvable for $k = 1$, which is the case for the problems we are interested in). The algorithm described at the beginning of the section shows that VERTEX COVER has a 2-way branching rule. Thus it seems that we have a simple framework for formally studying which problems can be solved by the technique of bounded-depth search trees.

Unfortunately, there are parameterized problems that do not have branching rules in the sense of Definition 4.1 for pathological reasons. For example, consider the (artificial) problem VERTEX COVER \uparrow defined as follows: given a graph G and an integer k , the task is to decide if G has a vertex cover of size k and, additionally, if $k = 2^i$ for some integer i (i.e., k is a power of 2). Clearly, this problem is not more complicated than VERTEX COVER: all we need is the trivial extra check whether k is a power of 2. Still, this problem has no branching rule:

Proposition 4.2. *Assuming $P \neq NP$, VERTEX COVER \uparrow does not have a branching rule.*

Proof. A simple padding argument shows that VERTEX COVER \uparrow is NP-hard. Suppose that \mathcal{A} is a branching algorithm for VERTEX COVER \uparrow that produces a constant number c of instances. We can assume that for every instance (I_i, k_i) created by \mathcal{A} , parameter k_i is a power of 2, since otherwise (I_i, k_i) is trivially a no-instance. Furthermore, we can assume that we run \mathcal{A} only on instances whose parameter is a power of 2. Therefore, if the parameter is 2^i , algorithm \mathcal{A} creates c instances with parameter at most 2^{i-1} . This means that the height of the search tree is at most $\log_2 k$ and therefore the size of the search tree is $O(c^{\log_2 k}) = O(k^{\log_2 c})$, which is polynomial in k (as c is a fixed constant). Thus we can solve VERTEX COVER \uparrow in polynomial time, implying $P = NP$. \square

To avoid situations like Proposition 4.2, we have to allow that a branching algorithm solves a modified version of the problem (e.g., VERTEX COVER instead of VERTEX COVER \uparrow). We express this by saying that we are interested in problems that can be reduced to a problem that has a branching rule. The right notion of reduction for this purpose is a restriction of parameterized reduction that runs in polynomial time and the parameter can be increased only by at most a constant factor:

Definition 4.3. *A linear-parameter polynomial-time parameterized transformation (LPPT) from a parameterized problem P_1 to a parameterized problem P_2 is a polynomial-time algorithm that, given an instance (I_1, k_1) of P_1 , creates an instance (I_2, k_2) of P_2 such that*

1. (I_1, k_1) is a yes-instance of P_1 if and only if (I_2, k_2) is a yes-instance of P_2 , and
2. $k_2 \leq c \cdot k_1$ for some constant c .

Now we can define the class BFPT (where B stands for “branching”), which formalizes the notion of branching:

Definition 4.4. *The class BFPT contains a parameterized problem P_1 if there is a parameterized problem P_2 that has a branching rule and there is an LPPT from P_1 to P_2 .*

Let us observe that VERTEX COVER \uparrow is in BFPT as expected: there is a trivial LPPT from this problem to VERTEX COVER.

Before discussing further examples of problems in BFPT, let us show a simple equivalent characterization of BFPT with linear-size witnesses. Recall that a language P is in NP if there is a polynomial-time decidable language P' and a polynomial p such that $x \in P$ if and only if there is a string w (the *witness*) of length at most $p(|x|)$ such that $(x, w) \in P'$. Informally, we can say that w is a polynomial-size witness for x that can be verified in polynomial. The following lemma shows that BFPT contains those NP languages where there is a witness whose length is linear in the parameter.

Lemma 4.5. *A parameterized problem is in BFPT if and only if there is a polynomial-time decidable language P' and a constant c such that $(x, k) \in P$ if and only if there is a string w of length at most $c|k|$ such that $(x, k, w) \in P'$.*

Proof. For the forward direction, suppose that parameterized problem P can be LPPT-reduced to a parameterized problem Q that has a c -way branching algorithm \mathcal{A} . Given an instance (I, k) of P , let (I', k') be the instance of Q created by the LPPT reduction. If $(I', k') \in Q$, then one of the branches of \mathcal{A} is successful, i.e., produced a yes-instance with parameter $k = 1$. As \mathcal{A} branches into c directions, we can describe with $\lceil \log_2 c \rceil \cdot k' = O(k)$ bits a successful branch. This description is a good witness for (I, k) : one can verify it in polynomial-time by computing the instance (I', k') given by the LPPT-reduction and then verifying that this branch of the search tree of \mathcal{A} is indeed successful.

For the reverse direction, let us define the language P'' such that $(x, k, w, \ell) \in P''$ if there is a string q of length at most ℓ such that $(x, k, wq) \in P'$. In other words, $(x, k, w, \ell) \in P''$ means that w can be extended with at most ℓ bits to a witness of (x, k) . The problem P'' parameterized by ℓ has a branching rule: we try to append a 0 or a 1 to w . Formally, $(x, k, w, \ell) \in P''$ if and only if either $(x, k, w) \in P'$ (which can be checked in polynomial time), or $(x, k, w0, \ell - 1) \in P''$, or $(x, k, w1, \ell - 1) \in P''$. By assumption, $(x, k) \in P$ if and only if there is a string w of length at most $c|k|$ such that $(x, k, w) \in P'$, or equivalently, $(x, k, \epsilon, c|k|) \in P''$ (where ϵ is the empty word). This gives an LPPT-reduction from P to P'' , a problem that has a branching algorithm. \square

Lemma 4.5 gives a more convenient way of showing that a problem is in BFPT. There are many examples of branching algorithms where the parameter does not necessarily decrease after each branching step, but we can show that some other measure strictly decreases. In such a case, it would be awkward to use directly the definition of BFPT, since we need to define an artificial problem where the parameter is the measure bounding the height of the search tree. On the other hand, with Lemma 4.5 all we need to do is to observe that we branch into

a constant number of directions in each step and the height of the search tree is bounded by a linear function of the parameter. Therefore, a string describing the successfully branch is a correct witness whose length is linear in the parameter.

As an example, let us use Lemma 4.5 to show that NODE MULTIWAY CUT (Given a graph G , a set of terminals $T \subseteq V(G)$, and an integer k , the task is to find a set S of at most k vertices that separates the terminals, that is, every component of $G \setminus T$ contains at most one vertex of T) is in BFPT. This problem is known to be FPT [78, 24, 31, 58]. Observation of, say, the $4^k \cdot n^{O(1)}$ algorithm of Chen et al. [24] shows that the search tree in the proof has height at most $2k$ and branching factor 2, thus there is a witness of $2k$ bits.

Proposition 4.6. NODE MULTIWAY CUT *is in* BFPT.

A standard technique in the design of parameterized algorithms is to solve the *compression* problem first. For example, let us consider the FEEDBACK VERTEX SET problem (given a graph G and an integer k , the task is to find a feedback vertex set of size k , that is, a set S of at most k vertices such that $G \setminus S$ is a forest). A randomized $4^k \cdot n^{O(1)}$ time algorithm was given in [8] and deterministic $2^{O(k \log k)} \cdot n^{O(1)}$ time algorithms were given already in [42, 11]. However, deterministic $c^k \cdot n^{O(1)}$ time algorithms appeared only much later and they all use the technique of compression [33, 59, 21, 30].

In the compression version of FEEDBACK VERTEX SET, the input contains additionally a feedback vertex set S_0 of size $k+1$. Intuitively, we have to “compress” a solution of size $k+1$ into a solution of size k . The compression problem can be easier than the original problem, as the initial solution S_0 can give us useful structural information about the graph. More generally, instead of starting with a solution having the specific size $k+1$, we can formulate the compression problem as starting with a solution of an arbitrary size $\ell > k$, and we parameterize by the problem by ℓ , the size of the initial solution.

There are two ways of using the compression algorithm to solve the original problem. The first method is to use the elegant technique of iterative compression, introduced by Reed et al. [92]. For a detailed explanation of this technique, see for example the survey [61]. The second method is to use a polynomial-time approximation algorithm to obtain a solution of size $f(k)$ and then use the compression algorithm to compress the initial solution of size $f(k)$ to a solution of size k (if such a solution exists). Let us observe that if we start with a constant-factor approximation and the compression is performed by a branching algorithm, then this combination yields a branching algorithm for the original problem. Therefore, we can state the following (somewhat informal) observation:

Proposition 4.7. *If a parameterized problem P has a polynomial-time constant-factor approximation and the compression version of P parameterized by the size of the initial solution is in BFPT, then P is in BFPT.*

FEEDBACK VERTEX SET has a 2-approximation and inspection of the proof, e.g., in [21] shows that the compression problem is in BFPT.

Proposition 4.8. FEEDBACK VERTEX SET *is in* BFPT.

There is an interesting connection between branching and kernelization. Suppose that a problem has a linear-vertex-kernel. Then the problem can be solved by computing the kernel and doing a brute force search on it. If this brute force search can be done by branching, then this gives a branching algorithm for the problem. We can formalize this by the following statement:

Proposition 4.9. *If a parameterized problem P admits a linear-vertex-kernel and the version of the problem parameterized by the number of vertices is in BFPT, then P is in BFPT.*

For example, this gives an alternate way of seeing that VERTEX COVER is in BFPT: it has a $2k$ -vertex-kernel [86] and VERTEX COVER parameterized by the number n of vertices can be trivially solved by branching, as it has a witness of n bits. Proposition 4.9 also applies to a wide range of planar problems. On planar graphs, many of the standard NP-hard problems become FPT and in fact admit linear-vertex-kernels; this follows for example from the powerful meta result of Bodlaender et al. [14]. For problems where the solution is a subset of vertices, it is usually trivial that the problem is FPT parameterized by the number of vertices, as a branching algorithm can enumerate all possible subsets. Therefore, we get for example the following results:

Proposition 4.10. INDEPENDENT SET, DOMINATING SET, CONNECTED DOMINATING SET, CONNECTED VERTEX COVER, INDUCED MATCHING *on planar graphs are in* BFPT.

However, let us note that Proposition 4.10 is somewhat unsatisfactory from the viewpoint of the optimality program. As these planar problems can be solved in time $c^{\sqrt{k}} \cdot n^{O(1)}$, it can be considered as irrelevant whether there are $c^k \cdot n^{O(1)}$ time branching algorithms for them.

MAX INTERNAL SPANNING TREE (given a graph G and an integer k , the task is to find a spanning tree where at least k vertices are non-leaves, that is, have degree more than one) admits a $3k$ -vertex-kernel, thus we might try to use Proposition 4.9 for this problem. However, it is not obvious if MAX INTERNAL SPANNING TREE parameterized by the number of vertices has a branching algorithm. A branching algorithm can guess the internal vertices, but then one has to enforce somehow that the degrees of these vertices are more than one. It is therefore an interesting open question whether the problem, parameterized by k or by the number of vertices, is in BFPT.

The example of MAX INTERNAL SPANNING TREE shows that the search for branching algorithms parameterized by the number n of vertices is also an interesting research question. This is particularly true for problems that can be solved in c^n time by dynamic programming techniques, for example, HAMILTONIAN PATH, CHROMATIC NUMBER, PARTITION INTO TRIANGLES for graphs having n vertices, SET PACKING over a universe of n elements, HITTING SET with n sets, etc. Paturi and Pudlák [89] raised a similar question: they ask if HAMILTONIAN PATH has a polynomial-time randomized algorithm with success

probability c^{-n} on n -vertex graphs. Note that if k -PATH parameterized by the length k of the path is in BFPT, then this implies that k -PATH parameterized by the number n of vertices is in BFPT, which further implies that HAMILTONIAN PATH has the required randomized algorithm: we can replace branching by random choices. This means that a negative answer to the question of Paturi and Pudlák would imply that k -PATH is not in BFPT. Therefore, if one wants to show that there is no such randomized algorithm, probably it makes sense to concentrate on first showing that k -PATH is not in BFPT, as this can be an easier question.

Branching algorithms are sometimes able to solve the more general counting version of the problem as well. This depends on the type of branching rule we use. If we know that every solution contains at least one element of a set S and we branch on the choice of exactly which subset of S is contained in the solution, then such a branching rule is usually good for counting: each solution remains a valid solution in exactly one of the branches, thus the number of solutions is exactly the sum of the number of solutions in all the branches. On the other hand, if we only know that whenever there is a solution, then there is also a solution containing an element of S and we branch on a subset of S , then this is typically not good for counting: we won't be able to count those solutions that are disjoint from S .

We could set up a framework for studying a stronger version of branching that is capable of solving counting problems. The main difference is that we want to require that the number of solutions is exactly the sum of the number of solutions in the different branches. We omit the details, as we do not have any interesting results at this point. The reason why we mention it, however, is the surprising fact that even though k -PATH is FPT, the counting version is known to be $\#W[1]$ -hard [49]. Thus it is unlikely that it is fixed-parameter tractable and hence unlikely that it has a branching rule suitable for counting. An interesting possibility is that one might be able to transfer this negative result on counting branching rules to ordinary branching rules solving the decision problem. It could be that understanding counting problems is the key for understanding branching.

Let us briefly mention that there is another natural question about branching: for a problem that can be solved by branching, what is the best branching algorithm? One way to formulate this question is to ask what the smallest c is such that there is a c -way branching algorithm for the problem. However, this c is always an integer by definition, but most of the sophisticated branching rules are asymmetric (i.e., different branches reduce the parameter by different values) and the analysis of such branching rules typically give a bound of c^k on the size of the search tree for some noninteger c . Therefore, probably it is a better and more relevant question to ask what the smallest c is such that the search tree is guaranteed to have size at most c^k . A different way to study the question is to find the smallest c such that there is a witness of size $c \cdot k$ for every instance. This question has been explored recently for SAT by Dantsin and Hirsch [32].

The aim of this section was to point out that the theoretical study of which problems can be solved by branching algorithms has been neglected so far and it is possible to study this question in a rigorous framework. We have formulated meaningful and challenging questions for future work. Let us conclude this section with a list of problems for which it would be interesting to decide if they are in BFPT:

- k -PATH parameterized by the length k of the path or by the number n of vertices.
- CONNECTED VERTEX COVER parameterized by the size k of the solution.
- STEINER TREE parameterized by the number k of terminals.
- HITTING SET parameterized by the number of sets.
- MAX INTERNAL SPANNING TREE parameterized by the number k of internal nodes in the tree.
- CHROMATIC NUMBER parameterized by the number n of vertices.

5 Problems on Directed Graphs

Finding algorithms for problems on directed graphs is typically more challenging than solving their undirected counterparts. Many of the tools for undirected graphs become more complicated to use or even break down completely when we move to the domain of directed graphs. This jump in complexity has been observed also in the context of fixed-parameter tractability. For example, the Graph Minors Theory of Robertson and Seymour was the early inspiration for parameterized complexity and many powerful results followed from it almost immediately. However, there is no directed analog of the theory and hence directed graph problems have not received the same initial boost that undirected problems have.

Despite the inherent difficulty of directed problems, there has been some progress in this direction. The most celebrated such result is the fixed-parameter tractability of the DIRECTED FEEDBACK VERTEX SET problem. The undirected FEEDBACK VERTEX SET problem was shown to be FPT already in 1992 [42] and subsequently several different algorithms have been found [11, 21, 33, 59]. The directed version (delete k vertices to make the graph acyclic) turned out to be much more challenging. It was not until 2008 that Chen et al. [25] proved the fixed-parameter tractability of DIRECTED FEEDBACK VERTEX SET via a clever combination of iterative compression and solving directed cut problems. Looking at the proof, one can observe that the algorithm is fairly elementary and in particular it does not use any deep results of Graph Minors Theory. Perhaps the reason why the resolution of this problem took so long was that people looked for inspiration at the wrong place: it was expected that the solution is very complex and would somehow follow from a generalization of graph structure theory to directed graphs. For example, the fixed-parameter tractability of FEEDBACK VERTEX SET follows immediately from standard treewidth techniques [11]. Directed analogs of treewidth do exist [64, 7, 71, 62, 9], but apparently they do not provide any help for this problem. In fact, there is some formal evidence that no

really useful directed width measure exists [56]. This means that when we are encountering directed problems, treewidth-based techniques, which are among the most useful theoretical tools in parameterized complexity, are missing from our arsenal. Recently, however, there were some attempts to build a useful structure theory for directed graphs from a very different direction by generalizing the undirected notion of nowhere dense graphs [72].

One of the most useful applications of bidimensionality theory [36]. This theory shows in a very easy way that subexponential-time parameterized algorithms exists for problems on planar and, more generally, on H -minor free graphs. It is a very interesting question whether subexponential-time algorithms follow with the same ease for directed planar problems. Dorn et al. [40] investigated this question, and gave $2^{O(\sqrt{k} \log k)} \cdot n^{O(1)}$ time algorithms for two problems on directed planar graphs, k -LEAF OUT BRANCHING and k -INTERNAL OUT BRANCHING. For both problems, the key is to use treewidth techniques on the underlying undirected graph: using problem specific arguments and reductions, large grids can be excluded or standard layering techniques can be made to work. It is also observed in [40] that the directed version of k -PATH for planar graphs can be solved in time $(1 + \epsilon)^k \cdot n^{f(\epsilon)}$ for every $\epsilon > 0$. That is, the base of the exponent can be made arbitrary close to 1 at the cost of increasing the exponent of n . Note that obtaining this running time is a weaker claim than having a $2^{o(k)} \cdot n^{O(1)}$ time algorithm. Therefore, it remains a very interesting open question if DIRECTED k -PATH on planar graphs can be solved in time $2^{o(k)} \cdot n^{O(1)}$, or perhaps even in time $2^{O(\sqrt{k})} \cdot n^{O(1)}$.

Due to the strong modeling power of graphs, sometimes graph problems appear in disguise. For example, ALMOST 2SAT is the problem of deciding whether the given 2SAT formula has an assignment satisfying all but at most k clauses. While graphs do not appear explicitly in the problem definition, the first FPT algorithm by Razgon and O'Sullivan [91] considers a natural directed graph formed by the implications and solves the problem by finding separators in this directed graph. In general, whenever the problem involves chains of implications, one can expect directed graphs to make an appearance.

Hard problems on undirected graphs are often studied on particular subclasses of graphs: on planar graphs, interval graphs, bounded-treewidth graphs, etc. It is natural to do the same when a problem turns out to be hard on general directed graphs. First, one can restrict the problem to directed graphs whose underlying undirected graph has special structure. For example, it is an important general question whether the well-understood techniques for planar undirected graphs (bidimensionality, Baker's layering approach, etc.) work for problems on planar directed graphs. Furthermore, there are interesting classes of directed graphs that have no undirected counterparts. The class of directed acyclic graphs seems to be an obvious choice to try: these graphs have useful properties (for example, dynamic programming on a topological ordering can be a useful approach), but still many problems are nontrivial on this class. Another well-studied class is the class of tournaments: directed complete graphs, i.e., there is exactly one directed edge between any two distinct vertices (one can also consider the more

general class of semicomplete directed graphs, where we allow bidirected edges as well). A classical result of Downey and Fellows [43] shows that DOMINATING SET is W[2]-complete for tournaments, that is, as hard as on general (directed) graphs. This is somewhat surprising in light of the fact that DOMINATING SET on tournaments can be solved in time $n^{O(\log n)}$ and is not known to be NP-hard. DIRECTED FEEDBACK ARC SET was known to be FPT on tournaments [90] even before its fixed-parameter tractability in general graphs [25] was shown, but recently it turned out that on tournaments there are subexponential-time FPT algorithms for the problem: it can be solved in time $2^{O(\sqrt{k})} \cdot n^{O(1)}$ [5, 46]. Problems related to edge- or vertex-disjoint paths in tournaments have been studied recently [53, 27, 28], but some interesting questions are still open: for example, the k vertex-disjoint paths problem is polynomial-time solvable for every fixed value of k , but it is not known to be fixed-parameter tractable.

A family of problems that received particular attention is formed by problems related to cuts and separation. These problems usually have both edge deletion and vertex deletion versions; for simplicity we discuss only the vertex deletion version here. Given a graph G , a set $T \subseteq V(G)$ of terminals, and an integer k , the NODE MULTIWAY CUT problem asks for a set of at most k vertices whose deletion separates the terminals from each other. The problem is known to be FPT by various techniques [78, 24, 31, 58]. A more general problem is MULTICUT, where the input contains a set of terminal pairs $(s_1, t_1), \dots, (s_\ell, t_\ell)$ and we have to disconnect s_i from t_i for every i . The fixed-parameter tractability of NODE MULTIWAY CUT implies in an easy way that there is an $f(k, \ell) \cdot n^{O(1)}$ time algorithm for the problem, i.e., it is FPT with combined parameters k and ℓ . Very recently, it was shown that MULTICUT is FPT even if parameterized by k only [16, 82]. The directed versions of these problems are very different and less understood. Unlike the undirected problem, DIRECTED MULTICUT is W[1]-hard parameterized by k [82]. However, the special case DIRECTED MULTIWAY CUT is FPT parameterized by k [26]. Somewhat surprisingly, the random sampling of important separators technique introduced in [82] for undirected MULTICUT works for the DIRECTED MULTIWAY CUT problem (but apparently it does not work for DIRECTED MULTICUT, as it is W[1]-hard). It remains an interesting open question whether DIRECTED MULTICUT is FPT with combined parameters k and ℓ . Note that the case $\ell = 2$ can be reduced to DIRECTED MULTIWAY CUT and hence is FPT by [26]. However, the case $\ell = 3$ is open. A possible motivation for the study of directed cut problems is that the solution of the DIRECTED FEEDBACK VERTEX SET problem [25] relied in large part on the fixed-parameter tractability of a variant of DIRECTED MULTICUT called SKEW MULTICUT, where we have to disconnect s_i from every t_j with $i \geq j$. It could be fruitful to investigate what other disconnection requirement patterns make the problem tractable. Most of these questions are meaningful also on directed acyclic graphs. Very recently, it has been proved that DIRECTED MULTICUT on directed acyclic graphs is fixed-parameter tractable with combined parameters k and ℓ , but W[1]-hard parameterized by k [70].

We conclude this section with two open questions that are easy to state but their solution would generalize and unify important known results. The first problem is DIRECTED EULERIAN DELETION (see [19, 29]): given a directed graph G and an integer k , delete k vertices such that every strongly connected component induces an Eulerian directed graph, that is, a graph where the indegree equals the outdegree for every vertex. It is not difficult to see that DIRECTED FEEDBACK VERTEX SET can be reduced to this problem, thus its solution should build on the arguments of [25]. The second problem is DIRECTED ODD CYCLE TRANSVERSAL: given a graph G and an integer k , delete k vertices such that there is no directed odd cycle in the remaining graph. Easy reductions show that this problem is more general than DIRECTED FEEDBACK VERTEX SET, ODD CYCLE TRANSVERSAL, and DIRECTED MULTIWAY CUT with $\ell = 2$. Thus a fixed-parameter tractability result for this problem would have to unify and generalize all the algorithmic ideas for these three problems.

References

1. Abu-Khzam, F.N.: An improved kernelization algorithm for r -set packing. *Inf. Process. Lett.* 110(16), 621–624 (2010)
2. Abu-Khzam, F.N.: A kernelization algorithm for d -hitting set. *J. Comput. Syst. Sci.* 76(7), 524–531 (2010)
3. Alber, J., Fernau, H., Niedermeier, R.: Parameterized complexity: exponential speed-up for planar graph problems. *J. Algorithms* 52(1), 26–56 (2004)
4. Alber, J., Fiala, J.: Geometric separation and exact solutions for the parameterized independent set problem on disk graphs. *J. Algorithms* 52(2), 134–151 (2004)
5. Alon, N., Lokshtanov, D., Saurabh, S.: Fast FAST. In: Albers, S., Marchetti-Spaccamela, A., Matias, Y., Nikolettseas, S., Thomas, W. (eds.) ICALP 2009, Part I. LNCS, vol. 5555, pp. 49–58. Springer, Heidelberg (2009)
6. Alon, N., Yuster, R., Zwick, U.: Color-coding. *J. ACM* 42(4), 844–856 (1995)
7. Barát, J.: Directed path-width and monotonicity in digraph searching. *Graphs and Combinatorics* 22(2), 161–172 (2006)
8. Becker, A., Bar-Yehuda, R., Geiger, D.: Randomized algorithms for the loop cutset problem. *J. Artif. Intell. Res. (JAIR)* 12, 219–234 (2000)
9. Berwanger, D., Dawar, A., Hunter, P., Kreutzer, S.: DAG-Width and Parity Games. In: Durand, B., Thomas, W. (eds.) STACS 2006. LNCS, vol. 3884, pp. 524–536. Springer, Heidelberg (2006)
10. Björklund, A.: Determinant sums for undirected hamiltonicity. In: Proceedings of the 51st Annual Symposium on Foundations of Computer Science, FOCS 2010, pp. 173–182 (2010)
11. Bodlaender, H.L.: On disjoint cycles. *Int. J. Found. Comput. Sci.* 5(1), 59–68 (1994)
12. Bodlaender, H.L.: Kernelization: New Upper and Lower Bound Techniques. In: Chen, J., Fomin, F.V. (eds.) IWPEC 2009. LNCS, vol. 5917, pp. 17–37. Springer, Heidelberg (2009)
13. Bodlaender, H.L., Downey, R.G., Fellows, M.R., Hermelin, D.: On problems without polynomial kernels. *J. Comput. Syst. Sci.* 75(8), 423–434 (2009)
14. Bodlaender, H.L., Fomin, F.V., Lokshtanov, D., Penninkx, E., Saurabh, S., Thilikos, D.M.: (Meta) kernelization. In: Proceedings of the 50th Annual Symposium on Foundations of Computer Science, FOCS 2009, pp. 629–638 (2009)

15. Bodlaender, H.L., Heggernes, P., Villanger, Y.: Faster parameterized algorithms for minimum fill-in. *Algorithmica* 61(4), 817–838 (2011)
16. Bousquet, N., Daligault, J., Thomassé, S.: Multicut is FPT. In: *Proceedings of the 43rd Annual Symposium on Theory of Computing, STOC 2011*, pp. 459–468 (2011)
17. Cai, L.: Fixed-parameter tractability of graph modification problems for hereditary properties. *Inform. Process. Lett.* 58(4), 171–176 (1996)
18. Calabro, C., Impagliazzo, R., Paturi, R.: The Complexity of Satisfiability of Small Depth Circuits. In: Chen, J., Fomin, F.V. (eds.) *IWPEC 2009*. LNCS, vol. 5917, pp. 75–85. Springer, Heidelberg (2009)
19. Cechlárová, K., Schlotter, I.: Computing the Deficiency of Housing Markets with Duplicate Houses. In: Raman, V., Saurabh, S. (eds.) *IPEC 2010*. LNCS, vol. 6478, pp. 72–83. Springer, Heidelberg (2010)
20. Chen, J., Fernau, H., Kanj, I.A., Xia, G.: Parametric duality and kernelization: Lower bounds and upper bounds on kernel size. *SIAM J. Comput.* 37(4), 1077–1106 (2007)
21. Chen, J., Fomin, F.V., Liu, Y., Lu, S., Villanger, Y.: Improved algorithms for feedback vertex set problems. *J. Comput. Syst. Sci.* 74(7), 1188–1198 (2008)
22. Chen, J., Huang, X., Kanj, I.A., Xia, G.: Linear FPT reductions and computational lower bounds. In: *Proceedings of the 36th Annual Symposium on Theory of Computing, STOC 2004*, pp. 212–221. ACM, New York (2004)
23. Chen, J., Kanj, I.A., Xia, G.: Improved upper bounds for vertex cover. *Theor. Comput. Sci.* 411(40–42), 3736–3756 (2010)
24. Chen, J., Liu, Y., Lu, S.: An Improved Parameterized Algorithm for the Minimum Node Multiway Cut Problem. In: Dehne, F., Sack, J.-R., Zeh, N. (eds.) *WADS 2007*. LNCS, vol. 4619, pp. 495–506. Springer, Heidelberg (2007)
25. Chen, J., Liu, Y., Lu, S., O’Sullivan, B., Razgon, I.: A fixed-parameter algorithm for the directed feedback vertex set problem. *J. ACM* 55(5) (2008)
26. Chitnis, R., Hajiaghayi, M., Marx, D.: Fixed-parameter tractability of directed multiway cut parameterized by the size of the cutset. In: *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012*, pp. 1713–1725 (2012)
27. Chudnovsky, M., Fradkin, A.O., Seymour, P.D.: Tournament immersion and cutwidth. *J. Comb. Theory, Ser. B* 102(1), 93–101 (2012)
28. Chudnovsky, M., Scott, A., Seymour, P.: Vertex disjoint paths in tournaments (manuscript)
29. Cygan, M., Marx, D., Pilipczuk, M., Pilipczuk, M., Schlotter, I.: Parameterized Complexity of Eulerian Deletion Problems. In: Kolman, P., Kratochvíl, J. (eds.) *WG 2011*. LNCS, vol. 6986, pp. 131–142. Springer, Heidelberg (2011)
30. Cygan, M., Nederlof, J., Pilipczuk, M., Pilipczuk, M., van Rooij, J.M.M., Wojtaszczyk, J.O.: Solving connectivity problems parameterized by treewidth in single exponential time. In: *Proceedings of the 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011*, pp. 150–159 (2011)
31. Cygan, M., Pilipczuk, M., Pilipczuk, M., Wojtaszczyk, J.O.: On Multiway Cut Parameterized above Lower Bounds. In: Marx, D., Rossmanith, P. (eds.) *IPEC 2011*. LNCS, vol. 7112, pp. 1–12. Springer, Heidelberg (2012)
32. Dantsin, E., Hirsch, E.A.: Satisfiability Certificates Verifiable in Subexponential Time. In: Sakallah, K.A., Simon, L. (eds.) *SAT 2011*. LNCS, vol. 6695, pp. 19–32. Springer, Heidelberg (2011)
33. Dehne, F., Fellows, M., Langston, M., Rosamond, F., Stevens, K.: An $O(2^{O(k)}n^3)$ FPT algorithm for the undirected feedback vertex set problem. *Theory Comput. Syst.* 41(3), 479–492 (2007)

34. Dell, H., Marx, D.: Kernelization of packing problems. In: Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, pp. 68–81 (2012)
35. Dell, H., van Melkebeek, D.: Satisfiability allows no nontrivial sparsification unless the polynomial-time hierarchy collapses. In: Proceedings of the 42nd Annual Symposium on Theory of Computing, STOC 2010, pp. 251–260 (2010)
36. Demaine, E.D., Hajiaghayi, M.: The bidimensionality theory and its algorithmic applications. *Comput. J.* 51(3), 292–302 (2008)
37. Demaine, E.D., Hajiaghayi, M., Thilikos, D.M.: The bidimensional theory of bounded-genus graphs. *SIAM J. Discrete Math.* 20(2), 357–371 (2006)
38. Demaine, E.D., Hajiaghayi, M.T.: Bidimensionality: new connections between FPT algorithms and PTASs. In: Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2005, pp. 590–601 (2005)
39. Dorn, F.: Planar subgraph isomorphism revisited. In: Proceedings 27th International Symposium on Theoretical Aspects of Computer Science, STACS 2010. Dagstuhl Seminar Proceedings, vol. 5, pp. 263–274. Leibniz-Zentrum für Informatik, Schloss Dagstuhl, Germany (2010)
40. Dorn, F., Fomin, F.V., Lokshtanov, D., Raman, V., Saurabh, S.: Beyond bidimensionality: Parameterized subexponential algorithms on directed graphs. In: Proceedings 27th International Symposium on Theoretical Aspects of Computer Science, STACS 2010, Dagstuhl Seminar Proceedings, vol. 5, pp. 251–262. Leibniz-Zentrum für Informatik, Schloss Dagstuhl, Germany (2010)
41. Downey, R.: A Basic Parameterized Complexity Primer. In: Bodlaender, H.L., et al. (eds.) *Fellows Festschrift*. LNCS, vol. 7370, pp. 91–128. Springer, Heidelberg (2012)
42. Downey, R.G., Fellows, M.R.: Fixed-parameter tractability and completeness. In: Proceedings of the Twenty-first Manitoba Conference on Numerical Mathematics and Computing, Winnipeg, MB, vol. 87, pp. 161–178 (1992)
43. Downey, R.G., Fellows, M.R.: Parameterized computational feasibility. In: Clote, P., Rémel, J. (eds.) Proceedings of the Second Cornell Workshop on Feasible Mathematics, Feasible Mathematics II, pp. 219–244. Birkhäuser, Boston (1995)
44. Downey, R.G., Fellows, M.R.: *Parameterized Complexity*. Monographs in Computer Science. Springer, New York (1999)
45. Eppstein, D.: Subgraph isomorphism in planar graphs and related problems. *J. Graph Algorithms Appl.* 3(3) (1999)
46. Feige, U.: Faster FAST (feedback arc set in tournaments). CoRR, abs/0911.5094 (2009)
47. Fellows, M.R., Fomin, F.V., Lokshtanov, D., Losievskaja, E., Rosamond, F.A., Saurabh, S.: Distortion Is Fixed Parameter Tractable. In: Albers, S., Marchetti-Spaccamela, A., Matias, Y., Nikolettas, S., Thomas, W. (eds.) *ICALP 2009, Part I*. LNCS, vol. 5555, pp. 463–474. Springer, Heidelberg (2009)
48. Fellows, M.R., Fomin, F.V., Lokshtanov, D., Rosamond, F.A., Saurabh, S., Szeider, S., Thomassen, C.: On the complexity of some colorful problems parameterized by treewidth. *Inf. Comput.* 209(2), 143–153 (2011)
49. Flum, J., Grohe, M.: The parameterized complexity of counting problems. *SIAM J. Comput.* 33(4), 892–922 (2004)
50. Flum, J., Grohe, M.: *Parameterized Complexity Theory*. Springer, Berlin (2006)
51. Fomin, F.V., Golovach, P.A., Lokshtanov, D., Saurabh, S.: Algorithmic lower bounds for problems parameterized with clique-width. In: Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, pp. 493–502 (2010)

52. Fomin, F.V., Kratsch, D.: *Exact Exponential Algorithms*, 1st edn. Springer (2010)
53. Fomin, F.V., Pilipczuk, M.: Jungles, bundles, and fixed parameter tractability. CoRR, abs/1112.1538 (2011)
54. Fomin, F.V., Villanger, Y.: Subexponential parameterized algorithm for minimum fill-in. In: *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012*, pp. 1737–1746 (2012)
55. Fortnow, L., Santhanam, R.: Infeasibility of instance compression and succinct PCPs for NP. In: *Proceedings of the 40th Annual Symposium on Theory of Computing, STOC 2008*, pp. 133–142 (2008)
56. Ganian, R., Hliněný, P., Kneis, J., Meister, D., Obdržálek, J., Rossmanith, P., Sikdar, S.: Are There Any Good Digraph Width Measures? In: Raman, V., Saurabh, S. (eds.) *IPEC 2010. LNCS*, vol. 6478, pp. 135–146. Springer, Heidelberg (2010)
57. Gramm, J., Niedermeier, R., Rossmanith, P.: Fixed-parameter algorithms for closest string and related problems. *Algorithmica* 37(1), 25–42 (2003)
58. Guillemot, S.: FPT algorithms for path-transversal and cycle-transversal problems. *Discrete Optimization* 8(1), 61–71 (2011)
59. Guo, J., Gramm, J., Hüffner, F., Niedermeier, R., Wernicke, S.: Compression-based fixed-parameter algorithms for feedback vertex set and edge bipartization. *J. Comput. System Sci.* 72(8), 1386–1396 (2006)
60. Hertli, T.: 3-SAT faster and simpler — Unique-SAT bounds for PPSZ hold in general. In: *Proceedings of the 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011*, pp. 277–284 (2011)
61. Hüffner, F., Niedermeier, R., Wernicke, S.: Techniques for practical fixed-parameter algorithms. *Comput. J.* 51(1), 7–25 (2008)
62. Hunter, P., Kreutzer, S.: Digraph measures: Kelly decompositions, games, and orderings. *Theor. Comput. Sci.* 399(3), 206–219 (2008)
63. Impagliazzo, R., Paturi, R., Zane, F.: Which problems have strongly exponential complexity? *J. Comput. System Sci.* 63(4), 512–530 (2001)
64. Johnson, T., Robertson, N., Seymour, P.D., Thomas, R.: Directed tree-width. *J. Comb. Theory, Ser. B* 82(1), 138–154 (2001)
65. Kaplan, H., Shamir, R., Tarjan, R.E.: Tractability of parameterized completion problems on chordal, strongly chordal, and proper interval graphs. *SIAM J. Comput.* 28(5), 1906–1922 (1999)
66. Kawarabayashi, K.: Planarity allowing few error vertices in linear time. In: *Proceedings of the 50th Annual Symposium on Foundations of Computer Science, FOCS 2009*, pp. 639–648 (2009)
67. Khot, S., Raman, V.: Parameterized complexity of finding subgraphs with hereditary properties. *Theor. Comput. Sci.* 289(2), 997–1008 (2002)
68. Khot, S., Regev, O.: Vertex cover might be hard to approximate to within $2 - \epsilon$. In: *18th Annual IEEE Conference on Computational Complexity (CCC 2003)*, pp. 371–378 (2003)
69. Kratsch, S.: Co-nondeterminism in compositions: A kernelization lower bound for a Ramsey-type problem. In: *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012*, pp. 114–122 (2012)
70. Kratsch, S., Pilipczuk, M., Pilipczuk, M., Wahlström, M.: Fixed-parameter tractability of multicut in directed acyclic graphs. CoRR, abs/1202.5749 (2012)
71. Kreutzer, S., Ordyniak, S.: Digraph decompositions and monotonicity in digraph searching. *Theor. Comput. Sci.* 412(35), 4688–4703 (2011)
72. Kreutzer, S., Tazari, S.: Directed nowhere dense classes of graphs. In: *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012*, pp. 1552–1562 (2012)

73. Langston, M.A.: Fixed-Parameter Tractability, A Prehistory. In: Bodlaender, H.L., Downey, R., Fomin, F.V., Marx, D. (eds.) *Fellows Festschrift*. LNCS, vol. 7370, pp. 3–16. Springer, Heidelberg (2012)
74. Lokshtanov, D., Marx, D., Saurabh, S.: Known algorithms on graphs of bounded treewidth are probably optimal. In: *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011*, pp. 777–789 (2011)
75. Lokshtanov, D., Marx, D., Saurabh, S.: Slightly superexponential parameterized problems. In: *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011*, pp. 760–776 (2011)
76. Lokshtanov, D., Misra, N., Saurabh, S.: Kernelization – Preprocessing with a Guarantee. In: Bodlaender, H.L., Downey, R., Fomin, F.V., Marx, D. (eds.) *Fellows Festschrift*. LNCS, vol. 7370, pp. 129–161. Springer, Heidelberg (2012)
77. Ma, B., Sun, X.: More efficient algorithms for closest string and substring problems. *SIAM J. Comput.* 39(4), 1432–1443 (2009)
78. Marx, D.: Parameterized graph separation problems. *Theoret. Comput. Sci.* 351(3), 394–406 (2006)
79. Marx, D.: On the optimality of planar and geometric approximation schemes. In: *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2007)*, pp. 338–348 (2007)
80. Marx, D.: Closest substring problems with small distances. *SIAM Journal on Computing* 38(4), 1382–1410 (2008)
81. Marx, D.: Can you beat treewidth? *Theory of Computing* 6(1), 85–112 (2010)
82. Marx, D., Razgon, I.: Fixed-parameter tractability of multicut parameterized by the size of the cutset. In: *Proceedings of the 43rd Annual Symposium on Theory of Computing, STOC 2011*, pp. 469–478 (2011)
83. Marx, D., Schlotter, I.: Obtaining a planar graph by vertex deletion. *Algorithmica* 62, 807–822 (2012)
84. Misra, N., Raman, V., Saurabh, S.: Lower bounds on kernelization. *Discrete Optimization* 8(1), 110–128 (2011)
85. Monien, B.: How to find long paths efficiently. In: *Analysis and design of algorithms for combinatorial problems (Udine, 1982)*. North-Holland Math. Stud, vol. 109, pp. 239–254. North-Holland, Amsterdam (1985)
86. Nemhauser, G.L., Trotter Jr., L.E.: Vertex packings: structural properties and algorithms. *Math. Programming* 8, 232–248 (1975)
87. Niedermeier, R.: *Invitation to fixed-parameter algorithms*. Oxford Lecture Series in Mathematics and its Applications, vol. 31. Oxford University Press, Oxford (2006)
88. Patrascu, M., Williams, R.: On the possibility of faster SAT algorithms. In: *Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010*, pp. 1065–1075 (2010)
89. Paturi, R., Pudlák, P.: On the complexity of circuit satisfiability. In: *Proceedings of the 42nd Annual Symposium on Theory of Computing, STOC 2010*, pp. 241–250 (2010)
90. Raman, V., Saurabh, S.: Parameterized algorithms for feedback set problems and their duals in tournaments. *Theor. Comput. Sci.* 351(3), 446–458 (2006)
91. Razgon, I., O’Sullivan, B.: Almost 2-SAT is fixed-parameter tractable. *J. Comput. Syst. Sci.* 75(8), 435–450 (2009)
92. Reed, B., Smith, K., Vetta, A.: Finding odd cycle transversals. *Operations Research Letters* 32(4), 299–301 (2004)

93. Robertson, N., Seymour, P.D.: Graph minors. XIII. The disjoint paths problem. *J. Combin. Theory Ser. B* 63(1), 65–110 (1995)
94. Robertson, N., Seymour, P.D.: Graph minors. XX. Wagner’s conjecture. *J. Combin. Theory Ser. B* 92(2), 325–357 (2004)
95. Scheffler, P.: A practical linear time algorithm for disjoint paths in graphs with bounded tree-width. Tech. Rep. 396/1994, Technical University of Berlin (1994)
96. Thilikos, D.M.: Graph Minors and Parameterized Algorithm Design. In: Bodlaender, H.L., Downey, R., Fomin, F.V., Marx, D. (eds.) *Fellows Festschrift. LNCS*, vol. 7370, pp. 228–256. Springer, Heidelberg (2012)
97. Williams, R.: Finding paths of length k in $O^*(2^k)$ time. *Inf. Process. Lett.* 109(6), 315–318 (2009)
98. Yap, C.K.: Some consequences of non-uniform conditions on uniform classes. *Theor. Comput. Sci.* 26, 287–300 (1983)

Author Index

- Bell, Tim 398
Betzler, Nadja 318
Bodlaender, Hans L. 196
Bredereck, Robert 318
- Casey, Nancy 398
Chen, Jianer 162
Chen, Jiehua 318
Chen, Yijia 364
- Downey, Rod 17, 91
- Flum, Jörg 364
Fomin, Fedor V. 457
- Gaspers, Serge 287
Gutin, Gregory 257
- Kanj, Iyad A. 162
Koblitz, Neal 39
- Langston, Michael A. 3
Lokshtanov, Daniel 129
- Marx, Dániel 457, 469
Misra, Neeldhara 129
- Niedermeier, Rolf 318
- Raman, Venkatesh 69
Rosamond, Frances 80, 398
- Saurabh, Saket 129
Stege, Ulrike 56
Szeider, Stefan 287
- Telle, Jan Arne 74
Thilikos, Dimitrios M. 228
- Wareham, Todd 51
Yeo, Anders 257