Rod Downey
Michael Fellows
Frank Dehne (Eds.)

LNCS 3162

Parameterized and Exact Computation

First International Workshop, IWPEC 2004
Bergen, Norway, September 2004
Proceedings

Springer

Rod Downey Michael Fellows
Frank Dehne (Eds.)

Parameterized and Exact Computation

First International Workshop, IWPEC 2004
Bergen, Norway, September 14-17, 2004
Proceedings

Springer

Volume Editors

Rod Downey
Victoria University, School of Mathematical and Computing Sciences
PO Box 600, Wellington, New Zealand
E-mail: Rod.Downey@mcs.vuw.ac.nz

Michael Fellows
The University of Newcastle, School of Electrical Engineering and Computer Science
Calaghan, NSW, Australia
E-mail: mfellows@cs.newcastle.edu.au

Frank Dehne
Griffith University, School of Computing and IT
Nathan, Brisbane, Qld 4111, Australia
E-mail: F.Dehne@griffith.edu.au

Library of Congress Control Number: 2004111137

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

ISSN 0302-9743
ISBN 3-540-23071-8 Springer Berlin Heidelberg New York

Springer is a part of Springer Science+Business Media

springeronline.com

Printed in Germany

Typesetting: Camera-ready by author, data conversion by Boller Mediendesign
Printed on acid-free paper SPIN: 11321361 06/3142 5 4 3 2 1 0

Lecture Notes in Computer Science 3162

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Preface

The central challenge of theoretical computer science is to deploy mathematics in ways that serve the creation of useful algorithms. In recent years there has been a growing interest in the two-dimensional framework of parameterized complexity, where, in addition to the overall input size, one also considers a parameter, with a focus on how these two dimensions interact in problem complexity.

This book presents the proceedings of the 1st International Workshop on Parameterized and Exact Computation (IWPEC 2004, `http://www.iwpec.org`), which took place in Bergen, Norway, on September 14–16, 2004. The workshop was organized as part of ALGO 2004. There were seven previous workshops on the theory and applications of parameterized complexity. The first was organized at the Institute for the Mathematical Sciences in Chennai, India, in September, 2000. The second was held at Dagstuhl Castle, Germany, in July, 2001. In December, 2002, a workshop on parameterized complexity was held in conjunction with the FST-TCS meeting in Kanpur, India. A second Dagstuhl workshop on parameterized complexity was held in July, 2003. Another workshop on the subject was held in Ottawa, Canada, in August, 2003, in conjunction with the WADS 2003 meeting. There have also been two Barbados workshops on applications of parameterized complexity.

In response to the IWPEC 2004 call for papers, 47 papers were submitted, and from these the program committee selected 25 for presentation at the workshop. In addition, invited lectures were accepted by the distinguished researchers Michael Langston and Gerhard Woeginger.

This first instantiation of a biennial workshop series on the theory and applications of parameterized complexity got its name in recognition of the overlap of the two research programs of *parameterized complexity* and *worst-case exponential complexity analysis*, which share the same formal framework, with an explicitly declared parameter of interest. There have been exciting synergies between these two programs, and this first workshop in the IWPEC series attempts to bring these research communities together.

The second workshop in this series is tentatively scheduled for the Gold Coast of Queensland, Australia, in July, 2006. An exact computation implementation challenge is being organized as a part of this second workshop. Details of the competition will be posted at `http://www.iwpec.org`.

On behalf of the program committee, we would like to express our appreciation to the invited speakers and to all authors who submitted papers. We also thank the external referees who helped with the process. We thank the program committee for excellent and thoughtful analysis of the submissions, and the organizers of ALGO 2004 in Bergen. We thank especially the tireless Frank Dehne for his efforts in almost all things relating to this conference and for co-editing these proceedings.

Rod Downey and Mike Fellows, July 2004

Organization

IWPEC Steering Committee

Jianer Chen
Frank Dehne
Rod Downey
Mike Fellows
Mike Langston
Rolf Niedermeier

IWPEC 2004 Program Committee

Rod Downey, co-chair
Michael Fellows, co-chair
Richard Beigel
Hans Bodlaender
Jianer Chen
Frank Dehne
Erik Demaine
Joerg Flum
Jens Gramm
Martin Grohe
Michael Hallett
Russell Impagliazzo
Michael Langston
Rolf Niedermeier
Mark Ragan
Venkatesh Raman
Peter Rossmanith
Jan Arne Telle
Dimitrios Thilikos
Gerhard Woeginger

Lecture Notes in Computer Science

For information about Vols. 1–3116

please contact your bookseller or Springer

Vol. 3241: D. Kranzlmüller, P. Kacsuk, J. Dongarra (Eds.), Recent Advances in Parallel Virtual Machine and Message Passing Interface. XIII, 452 pages. 2004.

Vol. 3240: I. Jonassen, J. Kim (Eds.), Algorithms in Bioinformatics. IX, 476 pages. 2004. (Subseries LNBI).

Vol. 3239: G. Nicosia, V. Cutello, P.J. Bentley, J. Timmis (Eds.), Artificial Immune Systems. XII, 444 pages. 2004.

Vol. 3238: S. Biundo, T. Frühwirth, G. Palm (Eds.), KI 2004: Advances in Artificial Intelligence. XI, 467 pages. 2004. (Subseries LNAI).

Vol. 3232: R. Heery, L. Lyon (Eds.), Research and Advanced Technology for Digital Libraries. XV, 528 pages. 2004.

Vol. 3224: E. Jonsson, A. Valdes, M. Almgren (Eds.), Recent Advances in Intrusion Detection. XII, 315 pages. 2004.

Vol. 3223: K. Slind, A. Bunker, G. Gopalakrishnan (Eds.), Theorem Proving in Higher Order Logic. VIII, 337 pages. 2004.

Vol. 3221: S. Albers, T. Radzik (Eds.), Algorithms – ESA 2004. XVIII, 836 pages. 2004.

Vol. 3220: J.C. Lester, R.M. Vicari, F. Paraguaçu (Eds.), Intelligent Tutoring Systems. XXI, 920 pages. 2004.

Vol. 3210: J. Marcinkowski, A. Tarlecki (Eds.), Computer Science Logic. XI, 520 pages. 2004.

Vol. 3208: H.J. Ohlbach, S. Schaffert (Eds.), Principles and Practice of Semantic Web Reasoning. VII, 165 pages. 2004.

Vol. 3207: L.T. Yang, M. Guo, G.R. Gao, N.K. Jha (Eds.), Embedded and Ubiquitous Computing. XX, 1116 pages. 2004.

Vol. 3206: P. Sojka, I. Kopecek, K. Pala (Eds.), Text, Speech and Dialogue. XIII, 667 pages. 2004. (Subseries LNAI).

Vol. 3205: N. Davies, E. Mynatt, I. Siio (Eds.), UbiComp 2004: Ubiquitous Computing. XVI, 452 pages. 2004.

Vol. 3203: J. Becker, M. Platzner, S. Vernalde (Eds.), Field Programmable Logic and Application. XXX, 1198 pages. 2004.

Vol. 3202: J.-F. Boulicaut, F. Esposito, F. Giannotti, D. Pedreschi (Eds.), Knowledge Discovery in Databases: PKDD 2004. XIX, 560 pages. 2004. (Subseries LNAI).

Vol. 3201: J.-F. Boulicaut, F. Esposito, F. Giannotti, D. Pedreschi (Eds.), Machine Learning: ECML 2004. XVIII, 580 pages. 2004. (Subseries LNAI).

Vol. 3199: H. Schepers (Ed.), Software and Compilers for Embedded Systems. X, 259 pages. 2004.

Vol. 3198: G.-J. de Vreede, L.A. Guerrero, G. Marín Raventós (Eds.), Groupware: Design, Implementation and Use. XI, 378 pages. 2004.

Vol. 3194: R. Camacho, R. King, A. Srinivasan (Eds.), Inductive Logic Programming. XI, 361 pages. 2004. (Subseries LNAI).

Vol. 3193: P. Samarati, P. Ryan, D. Gollmann, R. Molva (Eds.), Computer Security – ESORICS 2004. X, 457 pages. 2004.

Vol. 3192: C. Bussler, D. Fensel (Eds.), Artificial Intelligence: Methodology, Systems, and Applications. XIII, 522 pages. 2004. (Subseries LNAI).

Vol. 3190: Y. Luo (Ed.), Cooperative Design, Visualization, and Engineering. IX, 248 pages. 2004.

Vol. 3189: P.-C. Yew, J. Xue (Eds.), Advances in Computer Systems Architecture. XVII, 598 pages. 2004.

Vol. 3186: Z. Bellahsène, T. Milo, M. Rys, D. Suciu, R. Unland (Eds.), Database and XML Technologies. X, 235 pages. 2004.

Vol. 3185: M. Bernardo, F. Corradini (Eds.), Formal Methods for the Design of Real-Time Systems. VII, 295 pages. 2004.

Vol. 3184: S. Katsikas, J. Lopez, G. Pernul (Eds.), Trust and Privacy in Digital Business. XI, 299 pages. 2004.

Vol. 3183: R. Traunmüller (Ed.), Electronic Government. XIX, 583 pages. 2004.

Vol. 3182: K. Bauknecht, M. Bichler, B. Pröll (Eds.), E-Commerce and Web Technologies. XI, 370 pages. 2004.

Vol. 3181: Y. Kambayashi, M. Mohania, W. Wöß (Eds.), Data Warehousing and Knowledge Discovery. XIV, 412 pages. 2004.

Vol. 3180: F. Galindo, M. Takizawa, R. Traunmüller (Eds.), Database and Expert Systems Applications. XXI, 972 pages. 2004.

Vol. 3179: F.J. Perales, B.A. Draper (Eds.), Articulated Motion and Deformable Objects. XI, 270 pages. 2004.

Vol. 3178: W. Jonker, M. Petkovic (Eds.), Secure Data Management. VIII, 219 pages. 2004.

Vol. 3177: Z.R. Yang, H. Yin, R. Everson (Eds.), Intelligent Data Engineering and Automated Learning – IDEAL 2004. XVIII, 852 pages. 2004.

Vol. 3176: O. Bousquet, U. von Luxburg, G. Rätsch (Eds.), Advanced Lectures on Machine Learning. VIII, 241 pages. 2004. (Subseries LNAI).

Vol. 3175: C.E. Rasmussen, H.H. Bülthoff, B. Schölkopf, M.A. Giese (Eds.), Pattern Recognition. XVIII, 581 pages. 2004.

Vol. 3174: F. Yin, J. Wang, C. Guo (Eds.), Advances in Neural Networks - ISNN 2004. XXXV, 1021 pages. 2004.

Vol. 3173: F. Yin, J. Wang, C. Guo (Eds.), Advances in Neural Networks – ISNN 2004. XXXV, 1041 pages. 2004.

Vol. 3172: M. Dorigo, M. Birattari, C. Blum, L. M. Gambardella, F. Mondada, T. Stützle (Eds.), Ant Colony, Optimization and Swarm Intelligence. XII, 434 pages. 2004.

Vol. 3170: P. Gardner, N. Yoshida (Eds.), CONCUR 2004 - Concurrency Theory. XIII, 529 pages. 2004.

Vol. 3166: M. Rauterberg (Ed.), Entertainment Computing – ICEC 2004. XXIII, 617 pages. 2004.

Vol. 3163: S. Marinai, A. Dengel (Eds.), Document Analysis Systems VI. XI, 564 pages. 2004.

Vol. 3162: R. Downey, M. Fellows, F. Dehne (Eds.), Parameterized and Exact Computation. X, 293 pages. 2004.

Vol. 3160: S. Brewster, M. Dunlop (Eds.), Mobile Human-Computer Interaction – MobileHCI 2004. XVII, 541 pages. 2004.

Vol. 3159: U. Visser, Intelligent Information Integration for the Semantic Web. XIV, 150 pages. 2004. (Subseries LNAI).

Vol. 3158: I. Nikolaidis, M. Barbeau, E. Kranakis (Eds.), Ad-Hoc, Mobile, and Wireless Networks. IX, 344 pages. 2004.

Vol. 3157: C. Zhang, H. W. Guesgen, W.K. Yeap (Eds.), PRICAI 2004: Trends in Artificial Intelligence. XX, 1023 pages. 2004. (Subseries LNAI).

Vol. 3156: M. Joye, J.-J. Quisquater (Eds.), Cryptographic Hardware and Embedded Systems - CHES 2004. XIII, 455 pages. 2004.

Vol. 3155: P. Funk, P.A. González Calero (Eds.), Advances in Case-Based Reasoning. XIII, 822 pages. 2004. (Subseries LNAI).

Vol. 3154: R.L. Nord (Ed.), Software Product Lines. XIV, 334 pages. 2004.

Vol. 3153: J. Fiala, V. Koubek, J. Kratochvíl (Eds.), Mathematical Foundations of Computer Science 2004. XIV, 902 pages. 2004.

Vol. 3152: M. Franklin (Ed.), Advances in Cryptology – CRYPTO 2004. XI, 579 pages. 2004.

Vol. 3150: G.-Z. Yang, T. Jiang (Eds.), Medical Imaging and Augmented Reality. XII, 378 pages. 2004.

Vol. 3149: M. Danelutto, M. Vanneschi, D. Laforenza (Eds.), Euro-Par 2004 Parallel Processing. XXXIV, 1081 pages. 2004.

Vol. 3148: R. Giacobazzi (Ed.), Static Analysis. XI, 393 pages. 2004.

Vol. 3146: P. Érdi, A. Esposito, M. Marinaro, S. Scarpetta (Eds.), Computational Neuroscience: Cortical Dynamics. XI, 161 pages. 2004.

Vol. 3144: M. Papatriantafilou, P. Hunel (Eds.), Principles of Distributed Systems. XI, 246 pages. 2004.

Vol. 3143: W. Liu, Y. Shi, Q. Li (Eds.), Advances in Web-Based Learning – ICWL 2004. XIV, 459 pages. 2004.

Vol. 3142: J. Diaz, J. Karhumäki, A. Lepistö, D. Sannella (Eds.), Automata, Languages and Programming. XIX, 1253 pages. 2004.

Vol. 3140: N. Koch, P. Fraternali, M. Wirsing (Eds.), Web Engineering. XXI, 623 pages. 2004.

Vol. 3139: F. Iida, R. Pfeifer, L. Steels, Y. Kuniyoshi (Eds.), Embodied Artificial Intelligence. IX, 331 pages. 2004. (Subseries LNAI).

Vol. 3138: A. Fred, T. Caelli, R.P.W. Duin, A. Campilho, D.d. Ridder (Eds.), Structural, Syntactic, and Statistical Pattern Recognition. XXII, 1168 pages. 2004.

Vol. 3137: P. De Bra, W. Nejdl (Eds.), Adaptive Hypermedia and Adaptive Web-Based Systems. XIV, 442 pages. 2004.

Vol. 3136: F. Meziane, E. Métais (Eds.), Natural Language Processing and Information Systems. XII, 436 pages. 2004.

Vol. 3134: C. Zannier, H. Erdogmus, L. Lindstrom (Eds.), Extreme Programming and Agile Methods - XP/Agile Universe 2004. XIV, 233 pages. 2004.

Vol. 3133: A.D. Pimentel, S. Vassiliadis (Eds.), Computer Systems: Architectures, Modeling, and Simulation. XIII, 562 pages. 2004.

Vol. 3132: B. Demoen, V. Lifschitz (Eds.), Logic Programming. XII, 480 pages. 2004.

Vol. 3131: V. Torra, Y. Narukawa (Eds.), Modeling Decisions for Artificial Intelligence. XI, 327 pages. 2004. (Subseries LNAI).

Vol. 3130: A. Syropoulos, K. Berry, Y. Haralambous, B. Hughes, S. Peter, J. Plaice (Eds.), TeX, XML, and Digital Typography. VIII, 265 pages. 2004.

Vol. 3129: Q. Li, G. Wang, L. Feng (Eds.), Advances in Web-Age Information Management. XVII, 753 pages. 2004.

Vol. 3128: D. Asonov (Ed.), Querying Databases Privately. IX, 115 pages. 2004.

Vol. 3127: K.E. Wolff, H.D. Pfeiffer, H.S. Delugach (Eds.), Conceptual Structures at Work. XI, 403 pages. 2004. (Subseries LNAI).

Vol. 3126: P. Dini, P. Lorenz, J.N.d. Souza (Eds.), Service Assurance with Partial and Intermittent Resources. XI, 312 pages. 2004.

Vol. 3125: D. Kozen (Ed.), Mathematics of Program Construction. X, 401 pages. 2004.

Vol. 3124: J.N. de Souza, P. Dini, P. Lorenz (Eds.), Telecommunications and Networking - ICT 2004. XXVI, 1390 pages. 2004.

Vol. 3123: A. Belz, R. Evans, P. Piwek (Eds.), Natural Language Generation. X, 219 pages. 2004. (Subseries LNAI).

Vol. 3122: K. Jansen, S. Khanna, J.D.P. Rolim, D. Ron (Eds.), Approximation, Randomization, and Combinatorial Optimization. IX, 428 pages. 2004.

Vol. 3121: S. Nikoletseas, J.D.P. Rolim (Eds.), Algorithmic Aspects of Wireless Sensor Networks. X, 201 pages. 2004.

Vol. 3120: J. Shawe-Taylor, Y. Singer (Eds.), Learning Theory. X, 648 pages. 2004. (Subseries LNAI).

Vol. 3119: A. Asperti, G. Bancerek, A. Trybulec (Eds.), Mathematical Knowledge Management. X, 393 pages. 2004.

Vol. 3118: K. Miesenberger, J. Klaus, W. Zagler, D. Burger (Eds.), Computer Helping People with Special Needs. XXIII, 1191 pages. 2004.

Vol. 3117: M. Sonka, I.A. Kakadiaris, J. Kybic (Eds.), Computer Vision and Mathematical Methods in Medical and Biomedical Image Analysis. XII, 438 pages. 2004.

Table of Contents

Parameterized Enumeration, Transversals, and Imperfect Phylogeny Reconstruction*

Peter Damaschke

School of Computer Science and Engineering
Chalmers University, 41296 Göteborg, Sweden
ptr@cs.chalmers.se

Abstract. We study parameterized enumeration problems where we are interested in all solutions of limited size, rather than just some minimum solution. In particular, we study the computation of the transversal hypergraph restricted to hyperedges with at most k elements. Then we apply the results and techniques to almost-perfect phylogeny reconstruction in computational biology. We also derive certain concise descriptions of all vertex covers of size at most k in a graph, within less than the trivial time bound.

1 Introduction

We suppose familiarity with the notion of fixed-parameter tractable (FPT) problems, otherwise we refer to [8]. In many combinatorial optimization problems, one wants a particular solution where the parameter k is minimized. In the present paper we deal with the generation of *all* solutions with objective values bounded by parameter k. As a concrete application we study the reconstruction of almost perfect phylogenies.

A *perfect phylogeny (PP)* is a tree with nodes labeled by bit vectors of length m, and edges with labels from $[m] = \{1, \ldots, m\}$ such that, for every $i \in [m]$, the vectors having 0 and 1, respectively, at position i are separated by exactly one edge labeled i (and hence form connected subtrees). This is a fundamental structure in computational biology, as it describes evolutionary trees where at most one mutation appeared at every position. Another application domain is linguistics [26]. Recently, PP attracted new attention as it supports haplotype inference.[1]

The bit vectors are usually represented as rows of an $n \times m$ matrix. The columns correspond to the positions, also called sites or loci. We speak of a *PP matrix* if there is a PP containing all its rows (and perhaps more bit vecors) as node labels. From a PP matrix one can uniquely reconstruct such a PP in $O(nm)$ time. (Here, uniqueness means: subject to isomorphism and to the

* This work has been supported by a grant from the Swedish Research Council (Vetenskapsrådet), file no. 621-2002-4574.

[1] It is quite impossible to cite all relevant papers here. The reader is referred to the proceedings of RECOMB 2002-2004, including satellite workshops.

R. Downey, M. Fellows, and F. Dehne (Eds.): IWPEC 2004, LNCS 3162, pp. 1–12, 2004.

ordering of edge labels on paths of degree-2 nodes.) Reconstruction can be done *incrementally.* Starting from an empty set of columns, add columns successively to the input and refine the PP. Details are not complicated, see e.g. [27, Section 14.1]. One can generalize the notion of PP to non-binary cases, complexity results are in [1,2,19,25].

However, the PP assumption is often too strict. Repeated mutations at some loci, or recent immigration into a PP population leads to deviations from PP. Sequencing errors are also common, hence corrupted data may lose the PP property even if the true data would form a PP. Thus one should allow a small number k of changes, i.e. bit flips in the matrix, or extra rows or columns, or combinations of them. This motivates a few computational problems:

PP PLUS k ROWS: Given a binary matrix, find all sets of at most k rows the deletion of which leaves a PP matrix.
PP PLUS k COLUMNS: Similarly defined.
PP WITH k ERRORS: Given a binary matrix, find all sets of k bit flips such that the resulting matrix has a PP.

Enumerating all solutions captures the applications better than the minimization problem. There is no reason to assume that the smallest number of changes is always the correct explanation of data. Rather we want an overview of all consistent solutions, for at most k changes, and we also wish to reconstruct the part of the PP (i.e. without some rows or columns) common to all these conceivable solutions, the maximum agreement structure so to speak. Another phylogeny reconstruction problem has been studied in [14] from this perspective, see also [11] for more discussion of the importance of enumeration.

More generally (but a bit vaguely perhaps) it can be said that parameterized enumeration is suitable when we want to recognize certain objects from given data which do not perfectly fit the expected structure. Then all potential solutions are required for further inspection. Applications besides phylogeny may be found e.g. in data mining.

We will use a well-known characterization of PP matrices. A pair of columns is called *complete* if each of 00, 01, 10, 11 appears as a row in the submatrix induced by these two sites. Throughout the paper we refer to 00, 01, 10, 11 as *combinations.* The following has been discovered several times, see e.g. [15,28].

Theorem 1. *A matrix is a PP matrix iff it does not contain complete pairs.* □

This connects almost-PP reconstruction to the more abstract class of *subset minimization problems*: Given a set of n elements, a property π of subsets, and some k, we want all minimal subsets of size at most k enjoying π. Note carefully that the term *minimal* refers to set inclusion, not cardinality! We say that π is closed under $\supset$ if every $Y \supset X$ has property π whenever X has. Examples are vertex covers in graphs and hitting sets of set families (hypergraphs). For such π it suffices to know the minimal solutions, as they "represent" all solutions. This motivates the following

Definition 1. *Given a subset minimization problem, a* full kernel *is a set whose size depends on k only and contains all* minimal *solutions of size at most k.*

We call a problem inclusion-minimally fixed parameter enumerable (IMFPE) *if, for any instance of size n, all* minimal *solutions with value at most k are computable in time $O(f(k)p(n))$ where p is polynomial and f any function.*

Once we have a full kernel then, trivially, we can also enumerate the minimal solutions in time depending on k only, hence a problem is IMFPE in this case. It is crucial to notice the seemingly little but important difference to the optimally/minimally fixed parameter enumerable (MFPE) problems in [11]. To avoid confusion with minimum size, we added the attribute "inclusion-".

The family of all minimal hitting sets to a given set family is known as the *transversal hypergraph.* Applications include models of boolean formulae, database design, diagnosis, and data mining. Known algorithms for generating the transversal include a pseudo-polynomial output-sensitive algorithm [12], algorithms for special hypergraph classes [3], and a practical heuristic based on simple but powerful ideas [20]. Here we are interested in the "pruned" transversal hypergraph consisting of the minimal hitting sets of size at most k. Apparently, generation of hitting sets by ascending size has not been addressed before, unlike e.g. lexicographic ordering [18].

Contributions and organization of the paper: In Section 2 we obtain IMPFE results for subset minimization probnlems. In Section 3 we apply these findings to almost-PP reconstruction. In Section 4 we give an algorithm that outputs a certain concise description of all small vertex covers of a graph within less than the trivial time bound. Due to limited space, we defer the detailed exposition of results to these sections, and we could only sketch most proofs and only convey the main ideas.

We believe that the notions of IMFPE and a full kernel are more significant than some technical results which are built very much upon known research. (In particular, our results in Section 2 are close to [11], however, the new aspect is that the bounds still hold in the more demanding IMFPE setting.) The IMPFE concept is strict and sets limits to what clever algorithms could achieve, but as argued above, it seems to reflect the goals in certain applications well. Our focus is on theoretical results. Some experiments regarding the real performance on data of reasonable size would complete the picture. A handful open problems arise from the text.

More related literature: Recently, almost-PP reconstruction has also been studied in [28] in a more general frame (destroying all occurences of a given small submatrix), however without time bounds for enumeration. Results in [10] are based on a different, distance-based imperfection measure. The viewpoint in [26] is more similar to ours, but the focus was different, and exhaustive search is used for PP with extra columns. Various computational biology problems allow FPT results, see e.g. [9,13,14]. Closely related to error correction in PP matrices is reconstruction of PP from incomplete matrices [25,16]). It might be interesting to look at this NP-hard from the FPT point of view. Papers [23,24] contain results on directed PP reconstruction with missing entries. We mentioned maximum agreement problems (e.g. [17] gives an overview). Usually they have as input an arbitrary set of structures, rather than slight variants of one structure. In [6] we proposed a simple PP haplotyping algorithm for instances with enough genotypes, and the ideas in the present paper may lead to extensions to almost-PP populations.

2 Hitting All Small Hitting Sets

The VERTEX COVER problem is FPT [5,21]: Given a graph $G = (V, E)$ with n vertices and m edges, and a number k, find a k-vertex cover, i.e. a set of *at most* k vertices that is incident to every edge. A full kernel for VERTEX COVER is any subset of V that entirely contains all minimal k-vertex covers in G.

Lemma 1. VERTEX COVER *has a full kernel of size* $(1 + o(1))k^2$. *It can be constructed in* $O(m)$ *time.*

Proof. We show that the kernel from [4] is also a full kernel: Every k-vertex cover in G must contain the set H of vertices of degree larger than k. If we remove the vertices of H, all incident edges, and all vertices that had neighbors in H only, the remaining subgraph R has at most k^2 edges (or there is no solution at all), and hence less than $2k^2$ vertices. Every minimal k-vertex cover is the union of H and some minimal vertex cover of R. Thus, $H \cup R$ is a full kernel. Factor 2 can be improved to $1 + o(1)$ by more careful counting. (Omitted due to lack of space.) □

Remarks:

(1) For the optimization version of VERTEX COVER there exist kernels of size $2k$ [5], but $\Theta(k^2)$ is already the optimal worst-case bound for *full* kernels: In the disjoint union of m stars $K_{1,m}$ (one central vertex, joined to m leaves), the leaves of any star and the centers of all other stars build a k-vertex cover, $k = 2m - 1$. Hence the full kernel has size about $k^2/4$. The optimal constant in $\Theta(k^2)$ remains open.

(2) It was crucial to restrict the full kernel to *minimal* vertex covers. If we dropped the minmality condition, the size would not even be bounded by any function of k. A simple example is the star $K_{1,n-1}$ and $k = 2$: The center plus any leaf pair is a solution, and their union has size n. But the full kernel (merely the center) has size 1.

In order to enumerate all k-vertex covers we may construct the full kernel as and then apply the bounded search tree technique. Note that we distinguish *nodes* of the search tree from *vertices* of the graph.

Theorem 2. VERTEX COVER *is IMFPE. All minimal solutions of size at most* k *can be enumerated in* $O(m + k^2 2^k)$ *time.*

Proof. List all edges in the full kernel. Put a vertex from the first edge uv in the solution and branch for every choice (u or v). Label every new node by the vertex just selected. At any node proceed as follows: If some vertex in the edge listed next has already been chosen (i.e. it appears on the path from the root to the current node), then skip this edge. Repeat this step until the condition is false. Else, select a vertex from the next edge and branch.

Since this adds a new vertex to the solution on the considered tree path, but at most k vertices can be selected, the search tree has depth at most k, and at most 2^k leaves. Since every inner node has at least two children, the total size is $O(2^k)$. Finally we prune the tree, that is, successively remove all leaves where the edge list has not been scanned completely. From the search tree we can read off all k-vertex covers, as they are the label sets of paths from the root to the leaves. At every node we checked for every edge whether some of its vertices is already on the path. This gives immediately the time bound $O(k^2 2^k)$. Pruning costs $O(2^k)$ time.

One easiliy verifies that any minimal vertex cover X appears, in fact, as some path in the search tree.

Finally we also cut away leaves with non-minimal solutions X as follows. For every vertex in X, check whether all its neighbors are in X as well. Due to the degree bound in the kernel, this needs $O(k^2 2^k)$ time. □

HITTING SET: Given a hypergraph G with n vertices and h hyperedges (subsets of vertices), and a number k, find a set of at most k vertices that hits every hyperedge.

In c-HITTING SET, the cardinality of hyperedges is bounded by c, hence $c = 2$ is VERTEX COVER. For recent results on $c \geq 3$ see [22]. Next we study the enumeration version of an even more general problem. By a *multiedge* we mean a family of at most c disjoint sets. We omit c if it is clear from context. The following problem statement needed in 3.1 is quite natural as such and may be of independent interest, however we are not aware of earlier mention of it.

BOUNDED UNION: Given h multiedges, i.e. families of at most c disjoint sets, each with at most d vertices, find a subset U of at most k vertices, that *entirely includes* at least one set from each multiedge. In other words, find a union of sets, one from each multiedge, with size bounded by k.

We say that U *settles* a multiedge $\{S_1, \ldots, S_c\}$ if $S_i \subseteq U$ for some i. Thus, a solution to BOUNDED UNION must settle all multiedges. Note that HITTING SET is the special case when $d = 1$. On the other hand, BOUNDED UNION is trivially reducible to HITTING SET: Replace every multiedge $\{S_1, \ldots, S_c\}$ with the collection of all $|S_1| \times \ldots \times |S_c|$ hyperedges $\{s_1, \ldots, s_c\}$ such that $s_i \in S_i$ for $i = 1, \ldots, c$. Now, a set U hits all these hyperedges iff U settles the multiedge. It follows that this reduction also preserves all solutions. However, it blows up the input size by factor $O(d^c)$. Thus, one better works directly on instances of BOUNDED UNION, without the detour via this reduction.

Theorem 3. BOUNDED UNION *is IMFPE. All minimal solutions can be found in* $O(dc^{k+1}h + \min\{kc^{2k}, hkc^k\})$ *time.*

Proof. Again, we construct a bounded search tree, but now on the whole instance. List the given multiedges. Select a set from the first multiedge and branch for every choice. At any node proceed as follows: If the multiedge listed next is already settled by the union of previously selected sets on the tree path, then skip it. Repeat this step until the condition is false. Else, select a set from the next multiedge and branch. Since this adds at least one new element to the union, the search tree has depth at most k, at most c^k leaves, and $O(c^k)$ nodes in total. From the search tree we can read off all unions: In any path from the root to a leaf, collect the sets specified by the path. Completeness of the solution space can be easily established. As for the time bound, note that on each path, every multiedge is processed only once in $O(cd)$ time.

A naive method for filtering the non-minimal solutions is pairwise comparison in $O(kc^{2k})$ time. Testing the minimality of every solution X is faster if $h < c^k$. Proceed as follows. For every multiedge e, list the vertices of X contained in e. If exactly one set S of e satisfies $S \subseteq X$, then the vertices in S are not redundant. Mark all non-redundant vertices found that way. First suppose that all multiedges are already settled by these marked vertices. In this case, X is non-minimal iff X contains further, unmarked vertices. This check needs $O(hk)$ time. The other case is that some multiedges are not yet settled by the marked vertices. But since X is a solution, we conclude two things: (1) Not all vertices in X are marked. (2) For every multiedge, either one set consists of marked vertices only, or at least two sets are completely in X. Hence, we can remove an unmarked vertex from X, and still some set of every multiedge is in X. This means, X is not minimal, and we do not need further tests. □

We can show that a smaller *full* kernel exists in case $k > c$, thus generalizing a result from [22].

Theorem 4. *For any instance of* HITTING SET *or* BOUNDED UNION, *an equivalent instance with no more than* k^c *hyperedges can be obtained in time* $O(ck^{c-1}h)$. *Consequently, both problems have a full kernel of size* ck^c.

Proof. First we count how often every vertex appears in the hyperedges, in $O(cdh)$ time, going through the h hyperedges or multiedges. (For an instance of BOUNDED UNION, there is no need to perform the reduction to HITTING SET explicitly, as we know the cardinalities of sets in the multiedges.)

Suppose that each vertex appears in at most k^{c-1} hyperedges. Then, a set of size k can hit at most k^c hyperedges. If there is a solution at all, the instance contains only that many hyperedges, with a total of $k + (c-1)k^c$ vertices, and we are done. Otherwise we select a vertex and $k^{c-1} + 1$ hyperedges containing it.

Suppose by induction that we have found a set C of size i, and a family H_i of $k^{c-i} + 1$ hyperedges with C as subset. Either (1) some $C \cup \{y\}$, $y \notin C$ is in at least $k^{c-(i+1)} + 1$ hyperedges of H_i, or (2) k distinct vertices $y \notin C$ are not enough to hit all hyperedges of H_i. In case (1), the induction hypothesis holds for $i+1$. In case (2), each hitting set of size k must also hit C. But then we can create a hyperedge C and delete supersets of C in H_i from the instance, without altering the solution space.

This case distinction can be decided in $O((c-i)k^{c-i})$ time, since it suffices to consider all y from the union of members of H_i. We find the hyperedges in H_i that are to be deleted within the same time. If case (1) still holds for $i = c$, we have two copies of the same hyperedge and can also delete one. Altogether, we reduced the number of hyperedges, in $O(ck^{c-1})$ time.

The procedure is repeated less than h times. The vertex counters can be updated in time proportional to cd times the number of deleted hyperedges, which is a total of $O(cdh)$. Finally note that $d \leq k$ can be assumed. □

Combining the two results, we improve the coefficient of h from Theorem 3, provided that $k > c$:

Corollary 1. *All minimal solutions of* Bounded Union *can be computed in* $O(ck^{c-1}h + dc^{k+1}k^c + c^k k^{c+1})$ *time.*

Proof. Construct an instance that has the same solutions but at most k^c (rather than h) hyperedges, as in Theorem 4, then run the algorithm from Theorem 3 on it. □

3 Imperfect Phylogeny Reconstruction

3.1 Extra Rows

If an instance of PP plus k rows has a solution at all, then, in any complete pair, one of 00, 01, 10, 11 appears in at most k rows. At most 3 of these combinations appear at most k rows, unless $k \geq n/4$. In the following we implicitly assume $n > 4k$, remember that k is a fixed parameter. Destroying the complete pair means to remove all rows that contain *one of* 00, 01, 10, 11. This reduces PP plus k rows to Bounded Union: The rows of the matrix are elements of the ground set, and for every complete pair of columns i, j we define a multiedge whose sets are the sets of rows containing 00, 01, 10, 11, respectively, at sites i, j. Trivially, it is enough to keep sets of at most k rows. This gives $h \leq \binom{m}{2} < m^2$, $c = 3$, and $d = k$.

Before we state our theorem, we discuss a naive application of the Bounded Union results: Construct the multiedges from the matrix, then solve this instance of Bounded Union in $O(k^2m^2 + 3^k k^4)$ time (Corollary 1). To this end we may check all $O(m^2)$ column pairs for completeness. But, unfortunately, for each pair we have to look at almost all rows, thus preprocessing needs $O(nm^2)$ extra time. We get $O(nm^2 + 3^k k^4)$ and lose the benefits of a small kernel. By an idea mentioned in [16], the complete pairs of an $n \times m$ matrix can be found already in $O(nm^{\omega-1})$ time, where $O(n^\omega)$ is a bound for matrix multiplication. But still, the dependency in m is not linear. We omit any details, because the following time bound is anyhow an improvement, unless $3^k > m^{\omega-2}$.