Enumeration Algorithms in Data Profiling

Friedrich Martin Schirneck

Universitätsdissertation
zur Erlangung des akademischen Grades

Doctor rerum naturalium
(Dr. rer. nat.)

in der Wissenschaftsdisziplin
Theoretische Informatik

eingereicht an der
Digital Engineering Fakultät
der Universität Potsdam

Datum der Disputation: 16. Juni 2022

Betreuer

Prof. Dr. Tobias Friedrich
Hasso-Plattner-Institut, Universität Potsdam

Gutachter

Prof. Dr. Henning Fernau
Universität Trier

PD Dr. habil. Arne Meier
Leibniz Universität Hannover

For my wife, and also for myself.

Berlin, Fall 2021

Abstract

Data profiling is the extraction of metadata from relational databases. An important class of metadata are multi-column dependencies. They come associated with two computational tasks. The detection problem is to decide whether a dependency of a given type and size holds in a database. The discovery problem instead asks to enumerate all valid dependencies of that type. We investigate the two problems for three types of dependencies: unique column combinations (UCCs), functional dependencies (FDs), and inclusion dependencies (INDs).

We first treat the parameterized complexity of the detection variants. We prove that the detection of UCCs and FDs, respectively, is $\mathbf{W}[2]$-complete when parameterized by the size of the dependency. The detection of INDs is shown to be one of the first natural $\mathbf{W}[3]$-complete problems. We further settle the enumeration complexity of the three discovery problems by presenting parsimonious equivalences with well-known enumeration problems. Namely, the discovery of UCCs is equivalent to the famous transversal hypergraph problem of enumerating the hitting sets of a hypergraph. The discovery of FDs is equivalent to the simultaneous enumeration of the hitting sets of multiple input hypergraphs. Finally, the discovery of INDs is shown to be equivalent to enumerating the satisfying assignments of antimonotone, 3-normalized Boolean formulas.

In the remainder of the thesis, we design and analyze discovery algorithms for unique column combinations. Since this is as hard as the general transversal hypergraph problem, it is an open question whether the UCCs of a database can be computed in output-polynomial time in the worst case. For the analysis, we therefore focus on instances that are structurally close to databases in practice, most notably, inputs that have small solutions. The equivalence between UCCs and hitting sets transfers the computational hardness, but also allows us to apply ideas from hypergraph theory to data profiling. We devise an discovery algorithm that runs in polynomial space on arbitrary inputs and achieves polynomial delay whenever the maximum size of any minimal UCC is bounded. Central to our approach is the extension problem for minimal hitting sets, that is, to decide for a set of vertices whether they are contained in any minimal solution. We prove

that this is yet another problem that is complete for the complexity class **W**[3], when parameterized by the size of the set that is to be extended. We also give several conditional lower bounds under popular hardness conjectures such as the Strong Exponential Time Hypothesis (SETH). The lower bounds suggest that the running time of our algorithm for the extension problem is close to optimal.

We further conduct an empirical analysis of our discovery algorithm on real-world databases to confirm that the hitting set perspective on data profiling has merits also in practice. We show that the resulting enumeration times undercut their theoretical worst-case bounds on practical data, and that the memory consumption of our method is much smaller than that of previous solutions. During the analysis we make two observations about the connection between databases and their corresponding hypergraphs. On the one hand, the hypergraph representations containing all relevant information are usually significantly smaller than the original inputs. On the other hand, obtaining those hypergraphs is the actual bottleneck of any practical application. The latter often takes much longer than enumerating the solutions, which is in stark contrast to the fact that the preprocessing is guaranteed to be polynomial while the enumeration may take exponential time.

To make the first observation rigorous, we introduce a maximum-entropy model for non-uniform random hypergraphs and prove that their expected number of minimal hyperedges undergoes a phase transition with respect to the total number of edges. The result also explains why larger databases may have smaller hypergraphs. Motivated by the second observation, we present a new kind of UCC discovery algorithm called Hitting Set Enumeration with Partial Information and Validation (`HPIValid`). It utilizes the fast enumeration times in practice in order to speed up the computation of the corresponding hypergraph. This way, we sidestep the bottleneck while maintaining the advantages of the hitting set perspective. An exhaustive empirical evaluation shows that `HPIValid` outperforms the current state of the art in UCC discovery. It is capable of processing databases that were previously out of reach for data profiling.

Zusammenfassung

Data Profiling ist die Erhebung von Metadaten über relationale Datenbanken. Eine wichtige Klasse von Metadaten sind Abhängigkeiten zwischen verschiedenen Spalten. Für diese gibt es zwei wesentliche algorithmische Probleme. Beim Detektionsproblem soll entschieden werden, ob eine Datenbank eine Abhängigkeit eines bestimmt Typs und Größe aufweist; beim Entdeckungsproblem müssen dagegen alle gültigen Abhängigkeiten aufgezählt werden. Wir behandeln beide Probleme für drei Typen von Abhängigkeiten: eindeutige Spaltenkombinationen (UCCs), funktionale Abhängigkeiten (FDs) und Inklusionsabhängigkeiten (INDs).

Wir untersuchen zunächst deren parametrisierte Komplexität und beweisen, dass die Detektion von UCCs und FDs $\mathbf{W}[2]$-vollständig ist, wobei die Größe der Abhängigkeit als Parameter dient. Ferner identifizieren wir die Detektion von INDs als eines der ersten natürlichen $\mathbf{W}[3]$-vollständigen Probleme. Danach klären wir die Aufzählungskomplexität der drei Entdeckungsprobleme, indem wir lösungserhaltende Äquivalenzen zu bekannten Aufzählungsproblemen konstruieren. Die Entdeckung von UCCs zeigt sich dabei als äquivalent zum berühmten Transversal-Hypergraph-Problem, bei dem die Hitting Sets eines Hypergraphens aufzuzählen sind. Die Entdeckung von FDs ist äquivalent zum simultanen Aufzählen der Hitting Sets mehrerer Hypergraphen und INDs sind äquivalent zu den erfüllenden Belegungen antimonotoner, 3-normalisierter boolescher Formeln.

Anschließend beschäftigen wir uns mit dem Entwurf und der Analyse von Entdeckungsalgorithmen für eindeutige Spaltenkombinationen. Es ist unbekannt, ob alle UCCs einer Datenbank in worst-case ausgabepolynomieller Zeit berechnet werden können, da dies genauso schwer ist wie das allgemeine Transversal-Hypergraph-Problem. Wir konzentrieren uns daher bei der Analyse auf Instanzen, die strukturelle Ähnlichkeiten mit Datenbanken aus der Praxis aufweisen; insbesondere solche, deren Lösungen sehr klein sind. Die Äquivalenz zwischen UCCs und Hitting Sets überträgt zwar die algorithmische Schwere, erlaubt es uns aber auch Konzepte aus der Theorie von Hypergraphen auf das Data Profiling anzuwenden. Wir entwickeln daraus einen Entdeckungsalgorithmus, dessen Berechnungen auf beliebigen Eingaben nur polynomiellen Platz benötigen. Ist

zusätzlich die Maximalgröße der minimalen UCCs durch eine Konstante beschränkt, so hat der Algorithmus außerdem polynomiell beschränkten Delay. Der zentrale Baustein unseres Ansatzes ist das Erweiterbarkeitsproblem für minimale Hitting Sets, das heißt, die Entscheidung, ob eine gegebene Knotenmenge in einer minimalen Lösung vorkommt. Wir zeigen, dass dies, mit der Größe der Knotenmenge als Parameter, ein weiteres natürliches Problem ist, welches vollständig für die Komplexitätsklasse **W**[3] ist. Außerdem beweisen wir bedingte untere Laufzeitschranken unter der Annahme gängiger Schwere-Vermutungen wie der Starken Exponentialzeithypothese (SETH). Dies belegt, dass die Laufzeit unseres Algorithmus für das Erweiterbarkeitsproblem beinahe optimal ist.

Eine empirische Untersuchung unseres Entdeckungsalgorithmus auf realen Daten bestätigt, dass die Hitting-Set-Perspektive auch praktische Vorteile für das Data Profiling hat. So sind die Berechnungzeiten für das Finden der UCCs bereits sehr schnell und der Speicherverbrauch unseres Ansatzes ist deutlich geringer als der existierender Methoden. Die Untersuchung zeigt auch zwei interessante Verbindungen zwischen Datenbanken und ihren zugehörigen Hypergraphen: Einerseits sind die Hypergraphen, die alle relevanten Informationen enthalten, meist viel kleiner als die Eingabe-Datenbanken, andererseits ist die Berechnung dieser Hypergraphen die eigentliche Engstelle in der Praxis. Sie nimmt in der Regel viel mehr Zeit in Anspruch, als das Aufzählen aller Lösungen. Dies steht im deutlichen Gegensatz zu den bekannten theoretischen Resultaten, die besagen, dass die Hypergraph-Vorberechnung polynomiell ist, während der Aufzählungsschritt exponentielle Zeit benötigen kann.

Um die erste Beobachtung zu formalisieren, führen wir ein Maximum-Entropie-Modell für nicht-uniforme Hypergraphen ein und zeigen, dass die erwartete Anzahl ihrer minimalen Hyperkanten einen Phasenübergang druchläuft. Unsere Ergebnisse erklären auch warum größere Datenbanken mitunter kleinere Hypergraphen haben. Die zweite Beobachtung inspiriert uns zu einen Entdeckungsalgorithmus neuer Art, „Hitting Set Enumeration with Partial Information and Validation“ (`HPIValid`). Dieser nutzt die schnellen Aufzählungszeiten auf praktischen Daten aus, um die langwierige Berechnung des zu Grunde liegenden Hypergraphens zu beschleunigen. Dadurch umgehen wir die Engstelle und können gleichzeitig die Vorteile der Hitting-Set-Perspektive beibehalten. Eine ausgiebige empirische Analyse zeigt, dass `HPIValid` den aktuellen Stand der Technik im Bereich der UCC-Entdeckung deutlich übertrifft. `HPIValid` kann Datenbanken verarbeiten, für die Data Profiling zuvor unmöglich war.

Preface

This thesis presents the results of my research as a member of the Algorithm Engineering group, headed by Tobias Friedrich, at the Hasso Plattner Institute of the University of Potsdam from May 2015 until November 2021. Between January 2019 and June 2020, I was supported by the Investitionsbank des Landes Brandenburg (ILB) as part of the ILB ProFIT project "Virtual Compressor" under agreement number 80173319. Each chapter of this thesis is based on conference and journal publications with several coauthors. The thesis also contains some unpublished material, all of which is currently under review as part of the respective journal extensions. I give some context for the individual chapters below and highlight my contribution to the publications.

Chapter 3 After a talk by Michael Fellows (University of Bergen) at the Hasso Plattner Institute in September 2015, Felix Naumann approached Thomas Bläsius, Tobias Friedrich, and me to suggest studying multi-column dependency problems in relational databases through the lens of parameterized complexity. In the follow up, we were able to settle the complexity of detecting unique column combinations, functional dependencies, and inclusion dependencies. The results were jointly developed by Thomas Bläsius, Tobias Friedrich, and myself and published at the *11th International Symposium on Parameterized and Exact Computation* (IPEC 2016) [BFS16]. I contributed the $\mathbf{W}[2]$-completeness results and major parts of the writing. After publication, I recognized that essentially all results about the detection problems can be lifted to parsimonious equivalences for the corresponding discovery problems. For this thesis, I overhauled the exposition in order to adequately present the new results on enumeration. In the time between submission and defense, the chapter appeared in the *Theoretical Computer Science* journal [BFS22].

Chapter 4 Thomas Bläsius and I met Kitty Meeks (University of Glasgow) at IPEC 2016, where she presented her work on enumeration algorithms and extension problems. During her research visit in Potsdam in March 2017, we developed an algorithm for the transversal hypergraph problem. I proved its correctness and the $\mathbf{W}[3]$-completeness of the extension problem for minimal

hitting sets. These preliminary findings were further deepened at my return visit in Glasgow in May 2018. The same year, Thomas Bläsius and I co-advised the bachelor's thesis of Julius Lischeid who improved the upper bound on the delay and conducted extensive run time experiments. The combined theoretical and empirical work was announced at the *Dagstuhl Seminar on Algorithmic Enumeration* in October 2018 and appeared at the *21st Meeting on Algorithm Engineering and Experiments* (ALENEX 2019) [Blä+19b]. I was responsible for the write-up of both the conference version and the journal extension that appeared in the *Journal of Computer and System Sciences* [Blä+22]. For the latter, I proved several new fine-grained lower bounds. Erik Kohlros conducted additional experiments regarding the practical behavior of the extension oracle.

Chapter 5 This chapter originated as a section of the ESA 2020 paper [BFS20] (see Chapter 6), it gives a combinatorial proof for a version of the Chernoff–Hoeffding theorem that is tight up to constant factors. The presented results are entirely my own work.

Chapter 6 The empirical work on enumeration made apparent that often the number of minimal difference sets of a real-world database is much smaller than the size of the original input. Thomas Bläsius suggested a maximum-entropy model for hypergraphs to explain this behavior. I proved a phase transition in the expected size of its minimization. The main obstacle in the analysis were the loose bounds of the Chernoff–Hoeffding theorem, which had to be improved (see Chapter 5). The results were published at the *28th European Symposium on Algorithms* (ESA 2020) [BFS20].

Chapter 7 Since proposing the research topic to us, Felix Naumann and Thorsten Papenbrock continuously supported our work and shared their experiences from practice. This led to a joint collaboration whose preliminary results were announced at the *3rd International Workshop on Enumeration Problems and Applications* (WEPA) in October 2019. The final work appeared in the *Proceedings of the VLDB Endowment* [Bir+20] and were presented at the *46th International Conference on Very Large Databases* (VLDB 2020). For this paper, Johann Birnick, a bachelor student at that time, Thomas Bläsius, and I jointly developed the `HPIValid` algorithm. I proved a general result on the duality of hypergraphs with their transversals that implied the algorithm's correctness.

Besides the work that appears in this thesis, I was fortunate to be able to contribute also to other fields during my PhD studies. Building on my master thesis

as well as previous work [KP16], Timo Kötzing and I proposed the new research direction of compiling an atlas of computational learning theory [KS16]. Subsequently, this area has been further developed by myself [KSS17] and others [Ber+21; DK20; DK21a; DK21b; KKS21; KS21]. I also worked on the theory of evolutionary computation and randomized search heuristics in close collaboration with other researchers of the Algorithm Engineering group, colleagues from Frank Neumann's group at the University of Adelaide, and Benjamin Doerr (École Polytechnique). We showed that majority vote crossover [Fri+16] as well as island models with randomized rumor spreading [Doe+17; Doe+19] are powerful tools to accelerate evolutionary algorithms. We were also the first to rigorously investigate evolutionary computation under constraints [Fri+17; Fri+20; Fri+22; Shi+17; Shi+19; Shi+20]. Recently, I extended my interest to fault-tolerant data structures in a collaboration with Davide Bilò (University of L'Aquila) [Bil+21a; Bil+21b], Keerti Choudhary (Indian Institute of Technology Delhi) [Bil+22b], and other members of the Algorithm Engineering group [Bil+22a]. Finally, I co-advised several student research projects, among them were the master thesis of Philipp Fischbeck on reduction rules for the minimum hitting set problem [Blä+19a], and the bachelor thesis of Felix Mujkanovic [Muj+20] (preprint) on explainable time series classification.

A comprehensive list of all my publications, as of Summer 2022, can be found at the end of this thesis.

Acknowledgments

This thesis presents my research, but I want to express my gratitude to my wife, my parents, and my friends for continuously reminding me that there is more to life than that. Thank you Richard Hepting for explaining soccer to a football fan.

Research is teamwork. I thank my supervisor Tobias Friedrich, my HPI mentor Holger Giese, as well as Thorsten Papenbrock and Felix Naumann for suggesting data profiling as a research topic. I am extremely grateful to the Algorithm Engineering group for giving me an environment to flourish, contribute, and hone my skills. Thanks for the cake, board games, and ensuring me that I can turn to every one of you whenever I have a question (or to procrastinate). Shout-out to the A-1.13 crew Vanja Doskoč, Merlin de la Haye, Maximilian Katzmann, Ralf Rothenberger, and Ziena Zeif. You were kindly listening to my stories, solicited ones and others. I am indebted to the (former) students Johann Birnick, Erik Kohlros, and Julius Lischeid for their great engineering and coding abilities. The three colleagues that were most influential to me are Thomas Bläsius, Tobias, and Timo Kötzing. Thomas never lost his optimism, even when no idea we tried seemed to move us an inch. Every time I left his physical or virtual office, I felt a bit wiser than before. Tobias showed me the destination and then gave me the freedom to find my own path to it, occasionally reminding me of the waiting destination. Timo showed me how science is a passion and also a profession. Special thanks go to Thomas and Martin Krejca for providing the template for this thesis and Ralf for helping me adapting it to my needs. I also thank Martin for the many discussions about typography.

At last, I want to thank all my collaborators, who made my research not only possible but fun: Davide Bilò, Johann Birnick, Thomas Bläsius, Katrin Casel, Keerti Choudhary, Sarel Cohen, Benjamin Doerr, Vanja Doskoč, Philipp Fischbeck, Clemens Frahnow, Tobias Friedrich, Erik Kohlros, Timo Kötzing, Martin S. Krejca, J.A. Gregor Lagodzinski, Julius Lischeid, Kitty Meeks, Felix Mujkanovic, Samadhi Nallaperuma, Felix Naumann, Frank Neumann, Thorsten Papenbrock, Aishwarya Radhakrishnan, Patrick Schäfer, Leon Schiller, Karen Seidel, Feng Shi, Georg Tennigkeit, and Simon Wietheger.

Contents

1 Introduction

Data profiling is the extraction of metadata from relational databases. Among the first things a data scientist records when given a new dataset are the number of rows and columns, the value types in each column, like string, integer, float, or Boolean, the distribution of values, their range, sum, and mean. This information facilitates subsequent optimizations in storage and access and already provides valuable tools to unearth the knowledge in the data. A more subtle class of metadata are the hidden dependencies between values in multiple columns or even across different datasets. The prototype of multi-column dependencies are *unique column combinations* (UCCs). The table in Figure 1.1 gives an example. Knowing the `Name` and `Area Code` of an entry is already enough to identify any of the rows. The reason is that, say, the pair (“`Doe,John`”, “`UK-W1K`”) appears in the third row and the third row only. Note that neither the `Name` nor the `Area Code` alone have this discriminatory power, the UCC is *inclusion-wise minimal*. UCCs serve as small fingerprints of the whole database. They are natural candidates for primary keys, avoiding the need to introduce surrogate identification. More importantly, knowing the unique column combinations enables various data cleaning tasks as well as query optimization. The values appearing in a unique column combination are distinct by definition and form groups of size 1, thus SQL queries can skip the grouping phase and the `DISTINCT` operation, even if requested by the user. Also, the presence of UCCs allows for early returns of `SELECT` and `ORDER BY` operations. The surveys by Papenbrock, Naumann, and coauthors [Abe+18; KPN21] give an overview of many applications of UCCs.

Unfortunately, a dataset only rarely comes annotated with its dependencies. Much more often they need to be computed from raw data. This leads to two different computational tasks. The *detection* problem is to decide for a given database whether it admits a UCC with only a few columns. The *discovery* problem instead asks for a complete list of *all* minimal UCCs, regardless of their size and number. An equivalent term for the latter, which is probably more common in the algorithms community, is the *enumeration* of minimal solutions.

Age	Name	Address	City	Area Code
47	Mustermann, Max	Mittelstraße 125	Potsdam	D-14467
47	Mustermann, Max	W Broadway 400	San Diego	US-CA-92101
76	Doe, John	South Street 8	London	UK-W1K
90	Nightingale, Florence	South Street 8	London	UK-W1K
25	Menigmand, Morten	Trøjburgvej 24	Aarhus	DK-8200
33	Doe, John	South Street 8	Philadelphia	US-PA-19145

Figure 1.1: Example of a relational database. The column pair Name and Area Code is a minimal unique column combination.

In this thesis, we investigate those two problems for three types of multi-column dependencies: unique column combinations, functional dependencies (FDs), and inclusion dependencies (INDs). (A formal definition of the types is deferred to Chapter 2.) We examine the computational complexity of detection and discovery, and design enumeration algorithms and data models with a particular focus on applications in data profiling.

The discovery of data dependencies is intimately connected to the *transversal hypergraph* problem, which asks to enumerate the minimal hitting sets of a given hypergraph. Consider the first two rows of the database in Figure 1.1, they differ in the Address, City, and Area Code. Intuitively, any unique column combination must contain at least one of these three attributes as otherwise the two rows would be indistinguishable. It is a folklore result that the UCCs are indeed the hitting sets of the hypergraph of *difference sets* of all pairs of rows. The origins of the transversal hypergraph problem can be traced back to three independent papers all appearing in 1987 by Mannila and Räihä [MR87], Demetrovics and Thi [DT87], and, maybe lesser known, by Reiter [Rei87]. Interestingly, the first two articles derived the enumeration problem from database applications. Since then, finding the hitting sets of a hypergraph has also found uses in many other areas like artificial intelligence, machine learning, distributed systems, monotone logic, and bioinformatics. We refer the reader to the surveys by Eiter, Gottlob and coauthors [EG95; EMG08] as well as Gainer-Dewar and Vera-Licona [GV17].

The transversal hypergraph problem has developed into the most important yard stick for enumeration complexity. Unlike for decision problems, the border of tractability here does not run between polynomial and super-polynomial time, at least not when measured in the input size only. The number of solutions that need to be computed may be exponential, ruling out any polynomial algorithm.

Instead, one could hope for an algorithm that scales polynomially both in the input and the number of solutions. For more than three decades now, it is the major open question in enumeration whether the transversal hypergraph problem can be solved in *output-polynomial time*. The best known upper bound, given by Fredman and Khachiyan [FK96], is $N^{O(\log N/\log\log N)}$, where N is the combined input and output size. The fastest algorithm in practice, albeit without any performance guarantees, is the one by Murakami and Uno [GV17; MU14].

Eiter and Gottlob [EG95] showed that the unique column combinations of a database can be enumerated in output-polynomial time if and only if this is possible for the hitting sets of a hypergraph. This observation resulted from a line of research that examines enumeration problems indirectly through their "associated" decision problems. For example, the one corresponding to the transversal hypergraph problem is the decision, given two hypergraphs $\mathcal{H}$ and $\mathcal{G}$, whether $\mathcal{G}$ consists of exactly the minimal hitting sets of $\mathcal{H}$. Indeed, there is an output-polynomial algorithm for the transversal hypergraph problem if and only if the decision problem is solvable in polynomial time (in the combined sizes of $\mathcal{H}$ and $\mathcal{G}$) [BI95]. While those equivalences are theoretically pleasing, they are unusable in practice. The main issue is that data profiling algorithms based on such Turing-style reductions have a *space requirement* that scales with the number of solutions and is therefore inherently exponential. We instead relate the discovery problems more directly via so-called *parsimonious* reductions, without the indirection through a decision problem. This class of reductions is much more restrictive but, therefore, also more utile in practice. We generalize the equivalence of Eiter and Gottlob [EG95] by extending it to this class, and also obtain similar results for functional dependencies and inclusion dependencies.

While our results are entirely lifted to enumeration problems, our insights also stem from studying decision problems. However, we propose to use classes of problems that are much more closely related to enumeration itself. In the case of data profiling, the detection of unique column combinations, functional dependencies, and inclusion dependencies naturally also tells us something about the discovery of those objects. For the transversal hypergraph problem, the *extension* problem, deciding whether a partial solution is contained in any minimal hitting sets, turns out to be very fruitful. We approach those problems with a *parameterized analysis*. Conceived by Downey and Fellows [DF13; DF99], parameterized complexity is now a standard tool in the design and analysis of decision algorithms. It aims to ascribe the computational hardness to some

structural aspects of the inputs beyond their mere size. This is encompassed by a positive integer, the *parameter*, with a small value indicating "easy" instances. The hope is to find *fixed-parameter tractable* (FPT) algorithms for which the combinatorial explosion is confined to the parameter in such a way that, whenever it is constant, the running time is only polynomial in the input (with a degree independent of the parameter). Note that there exists a line of research, along the works of Fernau [Fer02], Damaschke [Dam06], Creignou et al. [Cre+17], Meier [Mei20], and Golovach et al. [Gol+22], that applies parameterized analysis also to enumeration complexity. In this thesis, we do not go into this direction for two reasons: First, we are primarily interested in classical (polynomial) enumeration complexity, only our tools are parameterized. Secondly, to obtain an enumeration algorithm with, say, FPT-delay it is often necessary that the associated decision problems are fixed-parameter tractable. This is not the case for us. In fact, all the decision problems we examine are surprisingly hard.

In the practitioner's eyes, **W**[1]-hardness is the parameterized analogue of **NP**-hardness. If a problem is hard for **W**[1] with a certain parameter, there is little hope to obtain an FPT-algorithm for it. From a complexity theoretician's perspective, however, **W**[1] is only the first level of an–presumably infinite–ladder of complexity classes known as the **W**-hierarchy. It has a class for each positive integer and beyond, all of which contain problems whose classical counterparts are in **NP**. Each of those classes is epitomized by their complete problems. Downey and Fellows [DF95a] defined **W**[t] as all parameterized problems reducible to the weighted satisfiability problem for Boolean circuits with of constant depth and weft t. (Again, see Chapter 2 for details.) While this family of problems is useful to reduce to, it is not very natural, that is, weft-t circuits do not regularly arise from practical computation. The classes **W**[1] and **W**[2] (and of course **FPT**, the 0-th level of the hierarchy) are populated with ample natural complete problems, most of them on graphs. Beyond the second level, however, only very few complete problems are known.

We add two more with this thesis. We show that the detection problem for inclusion dependencies, parameterized by the solution size, as well as the extension problem for minimal hitting sets, where the parameter is the size of the set to be extended, are both complete for the class **W**[3]. We thereby advance what is known about the medium levels of the **W**-hierarchy and develop new proof techniques to show hardness. This has already inspired new research at the intersection of parameterized complexity and data profiling. Prior to the

first announcement of the results presented here, there was only one natural **W**[3]-complete problem known, given by Chen and Zhang [CZ06] in the context of supply chain management. Since then, Casel et al. [Cas+21], building on Chapter 4 of this thesis, have shown **W**[3]-hardness already for the special case of extension to minimal dominating sets in bipartite graphs. Very recently, Hannula, Song, and Link [HSL21] used the ideas of Chapter 3 to prove that the detection of independence in databases is complete for **W**[3] as well.

The conjecture **FPT** ≠ **W**[1] can be seen as a conditional running time lower bound by ruling out FPT-algorithms for **W**[1]-hard problems, much in the same way as **P** ≠ **NP** does for **NP**-hard problems. The idea to derive barriers on the performance of efficient algorithms from plausible–albeit unproven–hardness assumptions has developed into a subfield of its own in recent years, called *fine-grained complexity* [Vas19]. It aims at pinpointing the exact exponent of the time needed to solve fundamental computational problems in the polynomial, exponential, as well as the parameterized domain. There are two especially popular conjectures in that area. The first one is the Exponential Time Hypothesis (ETH) [IP01], stating that the satisfiability of 3-CNF formulas on n variables cannot be decided in time $2^{o(n)}$. The other one is a strengthening of that, fittingly named Strong ETH (SETH) [IPZ01], positing that the same problem for CNF formulas with unbounded clause width cannot be solved in time $O(2^{(1-\varepsilon)n})$ for any constant $\varepsilon > 0$. Using those (and many more) hypotheses as starting points, a net of reductions has been developed that proves strong lower bounds for problems as diverse as finding the longest common subsequence of two strings, a maximum independent set in a graph, or deciding whether a Turing machine accepts its input within only a few steps. We apply ideas from fine-grained complexity to the extension problem for minimal hitting sets and offer several lower bounds. They differ both in the strength of the conjecture one is willing to assume and the resulting bound. All of them show that our solution for extension is close to optimal.

After settling the parameterized and enumeration complexity of the detection, extension, and discovery problems, we focus on the design and analysis of enumeration algorithms for unique column combinations. Since the close connection between data profiling and hitting sets is well-known for decades, it is surprising that it still bears some untapped potential for algorithmic improvements in both theory and practice. The connection does imply that discovering dependencies from data is computationally hard, which may discourage some practitioners.

Notwithstanding, we prove in this thesis that the hitting set perspective also enables new algorithms with drastically reduced space requirements and non-trivial performance guarantees on instances that occur in practice. Those are much needed in the era of big data. The most important obstacle holding back data profiling today is not the time needed for enumeration, but the large memory footprint of current methods. Usually, it is proportional to the number of solutions and once the outputs exceed main memory capacity, the resulting slow down stalls the whole profiling pipeline [Abe+18; PN17].

One of the prominent features of real-world databases is them having small multi-column dependencies, which we exploit algorithmically. Our first approach uses a search tree that is optimally pruned by an oracle for the extension problem for minimal hitting sets. It may seem counterintuitive to solve the enumeration problem by reducing it to a hard decision problem. The key insight is that the extension problem is tractable if the partial solution contains only a few vertices. Combined with the small solutions, we obtain an algorithm that uses linear space in the input only and, on m-edge, n-vertex hypergraphs with maximum solution size k^*, achieves a *delay*, the worst-case time between consecutive outputs, of $O(m^{k^*+1}n^2)$. Our algorithm is oblivious to k^*, which is *not* given as part of the input and is known to be **NP**-hard to compute or even approximate within a factor of $n^{1-\varepsilon}$ for any constant $\varepsilon > 0$ [Baz+18]. An experimental analysis shows that our enumeration method surpasses its worst-case running time guarantees on hypergraphs stemming from real-world databases and that the low memory footprint is a clear advantage of the hitting set perspective on data profiling.

However, the analysis also reveals two other phenomena. On the one hand, the hypergraphs of difference sets, which contain all information necessary for the enumeration, are often much smaller than the original input databases. On the other hand, while the enumeration succeeds within a reasonable time frame on practical data, the real bottleneck is the preprocessing of extracting those hypergraphs. Even though this computation is guaranteed to be polynomial, it usually takes much longer than the actual enumeration. The reason for both observations is that hardly any pair of rows contributes a meaningful, that is, inclusion-wise minimal difference set, and finding the relevant pairs is hard.

We investigate the size of the *minimization* of an hypergraph, that is, the number of its minimal edges, more rigorously by conducting an average-case analysis. Most of the existing models in the literature are for uniform hypergraphs, meaning that all edges have the same cardinality. Clearly, those are not suitable to

examine a property that stems exclusively from some edges being contained in others. We therefore introduce a maximum-entropy model for non-uniform hypergraphs. It turns out that the *perplexity* from information theory is the right concept to describe the size of the minimization in this model. While the more common *entropy function* measures the average number of bits needed to communicate the outcomes of a Bernoulli random variable, the perplexity is the exponential of the entropy. It is thus the total amount of contained information. Fitting the parameters of the variable to the size of the hypergraph results in a close estimate for the expected number of minimal edges.

We discover that this expectation undergoes a phase transition by showing that, up to the transition point, a constant fraction of all edges is minimal, but adding more edges beyond that causes the minimization to get *smaller* instead, until it eventually collapses to a single edge. Our results explain why we can expect only a few record pairs of a database to yield relevant information and why larger databases may have smaller corresponding hypergraphs. For the mathematical treatment, we need tail inequalities on the binomial distribution function that are tight up to constant factors. Such are usually derived from (a sufficiently tight variant of) the *Chernoff–Hoeffding theorem* [Hoe63]. We find that most of the existing so-called Chernoff bounds have some conceptual weaknesses. They are either too loose, exhibiting at least polynomial gaps, or their proofs use heavy analytic machinery and examine the binomial distribution only via its normal approximation. We dedicate the excursive Chapter 5 to give a combinatorial proof of Chernoff–Hoeffding that is tight up to constants.

Finally, we develop a completely new approach for the discovery of unique column combinations. It heavily exploits the second observation that enumeration is fast in practice while preprocessing the input database is slow. We call the algorithm *Hitting Set Enumeration with Partial Information and Validation* (`HPIValid`). Instead of receiving all difference sets up front, it works on a partial hypergraph (empty in the extreme case) and employs quick enumeration of candidate UCCs to pinpoint precisely the parts of the database where yet unseen information is hidden. We evaluate `HPIValid` on 54 databases from various application domains and compare it to the current state-of-the-art method in UCC discovery `HyUCC` [Pap+15; PN16; PN17]. `HPIValid` significantly improves over `HyUCC` with respect to both run time and memory usage. Those are direct consequences of viewing data profiling through the lens of hitting sets. In fact, the advantage of `HPIValid` over existing techniques is so extensive that it is able to process datasets that were previously out of reach for data profiling.

Contribution and Outline. In Chapter 3, we characterize the parameterized complexity of detecting unique column combinations, functional dependencies, and inclusion dependencies when parameterized by the solution size. We show that the problem for UCCs and FDs is $\mathbf{W}[2]$-complete, while it is $\mathbf{W}[3]$-complete for INDs. We derive the enumeration complexity of the corresponding discovery problems. Under parsimonious reductions, the discovery of UCCs is equivalent to the transversal hypergraph problem, while for FDs it is equivalent to the simultaneous enumeration of the hitting sets of multiple input hypergraphs. The discovery of maximal INDs is shown to be equivalent to enumerating the maximal satisfying assignments of antimonotone, 3-normalized Boolean formulas.

Chapter 4 shows the $\mathbf{W}[3]$-completeness of the extension problem for minimal hitting sets, with the size of the partial solution as parameter. We present an algorithm that decides whether a set X of vertices is extendable in an n-vertex, m-edge hypergraph in time $\mathrm{O}(m^{|X|+1}n)$. We give several fine-grained lower bounds. They show in their strongest form that, assuming SETH, the extension problem cannot be solved in time $m^{|X|-\varepsilon}\mathbf{poly}(n)$ for any constant $\varepsilon > 0$. We argue that closing the remaining gap of $\mathrm{O}(m)$ may be hard by showing that proving tight SETH-hardnesss violates a recently conjectured non-deterministic extension of SETH (NSETH) [Car+16]. We employ the extension algorithm as a subroutine to enumerate hitting sets and achieve linear space and a delay of $\mathrm{O}(m^{k^*+1}n^2)$, where k^* is the maximum size of any minimal solution. An experimental evaluation of the algorithm's performance on real-world data for the UCC discovery problem reveals that the preprocessing is the actual bottleneck in practice. Also, the corresponding hypergraphs are much smaller than the input database.

In Chapter 6, we introduce a maximum-entropy model for n-vertex, m-edge hypergraphs with expected edge size pn for $p \in (0, 1)$. Our main result is that, after $m = 1/(1-p)^{\alpha n}$ edges are sampled, the number of minimal edges is $\Theta(m) \cdot \mathrm{P}[\mathrm{Bin}(n, p) \leq (1-\alpha)n]$ in expectation. We then estimate this quantity in terms of the perplexity of the exponent α, which reveals a phase transition at $m^* = 1/(1-p)^{(1-p)n}$. We require a tail bound on the binomial distribution. For this purpose, Chapter 5 prepares a version of the Chernoff–Hoeffding theorem that is tight up to constant factors. We give a purely combinatorial proof.

In Chapter 7, we develop and engineer an algorithm for the discovery of unique column combinations working on partial input hypergraphs. We evaluate our method exhaustively on real-world data and show that it outperforms the current state of the art and broadens the reach of data profiling.

2 Preliminaries

This chapter introduces notation and basic concepts. We assume the reader to be familiar with discrete mathematics, probability theory, and classical (polynomial) complexity theory. For an overview on the latter, we recommend the textbook by Arora and Barak [AB09b]. For an exposition on probabilities, we refer the reader to Mitzenmacher and Upfal [MU17] or Motwani and Raghavan [MR95].

2.1 General Notation

Most of our notation is standard, we highlight only the specifics of this thesis. We let $\mathbb{N} = \{0, 1, 2, \dots\}$ denote the set of all non-negative integers, $\mathbb{N}^+ = \mathbb{N}\setminus\{0\}$ the positive integers, $\mathbb{R}_0^+$ the non-negative reals, and $\mathbb{R}^+ = \mathbb{R}_0^+\setminus\{0\}$ the positive reals. Unless explicitly stated otherwise, the variables i, j, k, ℓ range over $\mathbb{N}$, d, m, n range over $\mathbb{N}^+$, p, q, y, z over $\mathbb{R}_0^+$, and c, ε over $\mathbb{R}^+$. For some $d \in \mathbb{N}^+$, we let $[d] = \{1, 2, \dots, d\}$ denote the first d positive integers. We use $\vec{a}, \vec{b}, \vec{p}, \vec{x}$ to denote vectors, including infinite ones. For a suitable i, a_i is the i-th component of $\vec{a}$.

Sets are denoted by italic capitals A, B, E, F, H, R, S, T, U, V, X, Y, or Z. For a set S, $\mathcal{P}(S)$ is the power set of all subsets of S. Hypergraphs are subsets of the power set and are denoted by calligraphic capitals $\mathcal{D}$, $\mathcal{F}$, $\mathcal{G}$, $\mathcal{H}$, $\mathcal{S}$, $\mathcal{T}$, $\mathcal{U}$, and $\mathcal{W}$. They are our primary object of study, see Section 2.2 for more details

All computational objects are implicitly assumed to be encoded as finite bit strings, that is, elements of $\{0,1\}^* = \bigcup_{i\in\mathbb{N}}\{0,1\}^i$. We abstract from the technical details of this encoding and only assume that it is not excessive in length. In more detail, we are not concerned with the precise length of the resulting string and instead resort to more expressive complexity parameters. Those parameters are, however, required to be polynomially related to the length of the encoding. Taking hypergraphs as an example, we let n denote the number of vertices, and m the number of edges. Clearly, there is an encoding for hypergraphs that uses at most mn bits, is computable in time polynomial in $n + m$, and the hypergraph can be retrieved from the encoding in time polynomial in its bit-length.

We use big-O notation to describe growth rates in the usual way. We are mainly interested in the case where n tends to infinity. For example, we understand $\omega(1)$ as the class $\{f\colon \mathbb{N} \to \mathbb{R}_0^+ \mid \forall x \in \mathbb{R}_0^+ \, \exists n_0 \in \mathbb{N} \, \forall n \geq n_0 \colon f(n) \geq x\}$. Following convention, we use the non-symmetric equality sign to denote membership, that is, $n = \mathrm{O}(n \log n)$ instead of $n \in \mathrm{O}(n \log n)$. If the input size is additionally described by some parameter m, it is implicitly assumed to be a function of n and only the univariate asymptotics in n, for arbitrary choices of $m = m(n)$, are expressed. Two notable exception to this rule are the running time of output-polynomial enumeration algorithms (see Section 2.4.1) as well as the complexity of parameterized decision problems and algorithms (Section 2.4.2). We let $\mathbf{poly}(n) = \bigcup_{i \in \mathbb{N}} \mathrm{O}(n^i)$ denote the class of polynomial growth rates. For any m, the notation $\mathbf{poly}(n, m)$ stands for $\mathbf{poly}(n + m)$, analogously for more than two arguments.

2.2 Hypergraphs and Hitting Sets

This thesis is about hypergraphs. The concepts introduced in this section are taken from [Ber89]. A *hypergraph* is a non-empty, finite *vertex set* $V \neq \emptyset$ together with a set of subsets $\mathcal{H} \subseteq \mathcal{P}(V)$, the *(hyper-)edges*. We identify a hypergraph with its edge set $\mathcal{H}$ if this does not create any ambiguities. We do not exclude special cases of this definition like the empty graph ($\mathcal{H} = \emptyset$), an empty edge ($\emptyset \in \mathcal{H}$), or isolated vertices ($V \supsetneq \bigcup_{E \in \mathcal{H}} E$). We let $n = |V|$ denote the number of vertices and $m = |\mathcal{H}|$ the number of edges. The *rank* of a hypergraph $\mathcal{H}$ is its maximum edge cardinality, $\operatorname{rank}(\mathcal{H}) = \max_{E \in \mathcal{H}} |E|$. If all edges have the same size, the hypergraph is *uniform*. A *graph* is a uniform hypergraph of rank 2. We do not prescribe any a priori bound on the rank; in particular, the hypergraphs we deal with are usually neither uniform nor is their rank bounded.

A hypergraph is *Sperner*[1] if none of its edges is contained in another. The *minimization* of $\mathcal{H}$ is the subset of all inclusion-wise minimal edges, $\min(\mathcal{H}) = \{E \in \mathcal{H} \mid \forall E' \in \mathcal{H} \colon E' \subseteq E \Rightarrow E' = E\}$. This should not be confused with the notation for a minimum element of a set. The minimization is always Sperner.

In *multi-hypergraphs*, we allow multiple copies of the same edge to be present, that is, $\mathcal{H}$ is a multiset. However, we never allow multiple copies of the same vertex. The definition of the rank immediately transfers to multi-hypergraphs;

1 Berge [Ber89] calls Sperner hypergraphs *simple*. This term is not used consistently in the literature, compare [FM13].

the notion of Sperner hypergraphs is not applicable. To transfer the minimization, we require that whenever a minimal edge has multiple copies, only one of them is included in the minimization. This way, $\min(\mathcal{H})$ is always a mere hypergraph (a set). For a multi-hypergraph $\mathcal{H}$, we use $|\mathcal{H}|$ to denote the total number of edges counting multiplicities, and $\|\mathcal{H}\|$ for the number of distinct edges. Evidently, we have $|\min(\mathcal{H})| \leq \|\mathcal{H}\| \leq |\mathcal{H}|$.

Any total ordering $\preccurlyeq$ of the vertex set V induces a *lexicographical order* on the subsets of V. For two sets $S, T \subseteq V$, let $S \triangle T = (S \setminus T) \cup (T \setminus S)$ denote the symmetrical difference. Set S is *lexicographically smaller*[2] than T, denoted $S \preccurlyeq_{\text{lex}} T$, if $S = T$ or $\min_{\preccurlyeq}(S \triangle T) \in S$. This means that the $\preccurlyeq$-first element in which S and T differ is in S. We occasionally call a hypergraph over a totally ordered vertex set an *ordered hypergraph*.

We define a preorder[3] $\preccurlyeq$ on hypergraphs by requiring that, for hypergraphs $\mathcal{H}, \mathcal{G}$ on the same vertex set, $\mathcal{H} \preccurlyeq \mathcal{G}$ shall hold if every edge of $\mathcal{H}$ contains some edge of $\mathcal{G}$. This generalizes set inclusion as $\mathcal{H} \subseteq \mathcal{G}$ implies $\mathcal{H} \preccurlyeq \mathcal{G}$. On Sperner hypergraphs, $\preccurlyeq$ is antisymmetric. Namely, it is then a partial order.

A *hitting set*, or *transversal*, for a hypergraph $(V, \mathcal{H})$ is a set $T \subseteq V$ of vertices such that T has a non-empty intersection with every edge $E \in \mathcal{H}$. A hitting set is *(inclusion-wise) minimal* if it does not contain any other hitting set. We make extensive use of the following well-known observation, which is an easy consequence of the definition of minimality.

▶ **Observation 2.1 (Folklore).** Let $(V, \mathcal{H})$ be a hypergraph. A hitting set $T \subseteq V$ for $\mathcal{H}$ is minimal if and only if for every element $x \in T$, there exists an edge $E_x \in \mathcal{H}$ such that $E_x \cap T = \{x\}$. ◀

We call such an E_x a *private (hyper-)edge*[4] of x with respect to the transversal T.

The minimal hitting sets of some hypergraph $\mathcal{H}$ form a Sperner hypergraph on the same vertex set, the *transversal hypergraph* $\mathrm{Tr}(\mathcal{H})$. We denote its cardinality by $N_{\min} = |\mathrm{Tr}(\mathcal{H})|$. We use the term *transversal rank* of $\mathcal{H}$ to refer to the maximum cardinality of its minimal hitting sets, that is, $\mathrm{rank}(\mathrm{Tr}(\mathcal{H}))$. If there is no ambiguity about the underlying hypergraph, we abbreviate it as k^*. Regarding transversals, it does not make a difference whether the full hypergraph $\mathcal{H}$

2 The definition follows [JPY88]. The same concept has also been described as the lexicographically *larger* set, compare [EGM03].
3 A *preorder* is a binary relation that is reflexive and transitive.
4 Private edges are also known as *critical hyperedges* [MU14].

is considered or its minimization, it holds that $\mathrm{Tr}(\min(\mathcal{H})) = \mathrm{Tr}(\mathcal{H})$. The minimization and the transversal hypergraph are mutually dual in the sense that the minimal edges are exactly the minimal hitting sets of the minimal hitting sets, meaning $\mathrm{Tr}(\mathrm{Tr}(\mathcal{H})) = \min(\mathcal{H})$.

2.3 Relational Databases

Most of the hypergraphs we discuss arise from relational databases. Our notation of the latter is an adaption of the one used by Abedjan et al. [Abe+18]. A *(relational) schema* R is a non-empty, finite set of *attributes* or *columns*. Each attribute comes implicitly associated with a set of admissible values. A *row*, or *record*, over schema R is a tuple r whose entries are indexed by R such that, for any attribute $a \in R$, the value $r[a]$ is admissible for a. A *(relational) database* $\mathfrak{r}$ over R is a finite set of such records. The database and the schema will usually be clear from the context. For any row r and any subset $X \subseteq R$ of columns, $r[X]$ is the subtuple of r projected onto S. We let $\mathfrak{r}[X]$ denote the family of all such projections of rows in $\mathfrak{r}$. Note that $\mathfrak{r}[X]$ is a multiset as the same combination of values may appear in different rows.

2.3.1 Unique Column Combinations

A *unique column combination* (UCC), or simply *unique*, for some database $\mathfrak{r}$ over schema R, is a subset $U \subseteq R$ of attributes such that for any two records $r, s \in \mathfrak{r}$, $r \neq s$, there is an attribute $a \in U$ such that $r[a] \neq s[a]$. Intuitively, a UCC U uniquely identifies the rows of a database as any combination of values of attributes in U appears at most once among the rows, the collection $\mathfrak{r}[U]$ is indeed a set (as opposed to a multiset). We say a UCC U' is a *generalization* of a UCC U if $U' \subsetneq U$. A UCC is *(inclusion-wise) minimal* if it does not contain any other UCC, that is, if it does not have a generalization.

There is an intimate connection between UCCs and transversals in hypergraphs. To describe it, we need the notion of difference sets. For any two distinct rows r_1, r_2, $r_1 \neq r_2$, over the same schema R, their *difference set* $\{a \in R \mid r_1[a] \neq r_2[a]\}$ is the set of attributes in which the rows disagree. For a fixed database $(R, \mathfrak{r})$, a difference set is *(inclusion-wise) minimal* if it does not properly contain a difference set for another pair of rows in $\mathfrak{r}$. We denote the hypergraph of minimal difference sets by $(R, \mathcal{D})$. The following observation is well-known in

the literature, see [Abe+18; Dat03]. Probably the first explicit mention was by Mannila and Räihä [MR87]. While the proof is elementary, we include it here due to the result's fundamental importance for this thesis.

▶ **Observation 2.2 (Folklore).** The unique column combinations are the hitting sets of difference sets. In particular, let $\mathfrak{r}$ be a database and $\mathcal{D}$ the hypergraph of its minimal difference sets. Then, any edge of $\mathrm{Tr}(\mathcal{D})$ is a minimal UCC of $\mathfrak{r}$ and there are no other minimal UCCs. ◀

Proof. It is sufficient to show that any unique of $\mathfrak{r}$ is a hitting set for $\mathcal{D}$ and vice versa. The assertion of minimality then follows from the definition of $\mathrm{Tr}(\mathcal{D})$. Conversely, any (non-minimal) UCC/hitting set can be generated from a minimal one by adding more attributes/vertices arbitrarily.

Let R be the schema of $\mathfrak{r}$ and $C \subseteq R$ a set of attributes. If there exists an edge $E \in \mathcal{D}$ such that $C \cap E = \emptyset$, then there are two rows $r, s \in \mathfrak{r}$ such that their difference set is E and C does not contain any attribute in which r and s disagree. Now suppose C is a hitting set for $\mathcal{D}$, that is, it contains an attribute of every minimal difference set. Clearly, C then intersects all difference sets (not only the minimal ones) of pairs of rows in $\mathfrak{r}$, it is a UCC. ■

2.3.2 Null Semantics

Relational data in practice may exhibit a `null` marker $\perp$, distinct from all admissible values, to indicate that there is no value in the corresponding entry [GUW08]. This is an issue for the discovery of unique column combinations, because the difference sets may vary depending on how `null` markers compare to the values and other `null` markers. In all our empirical evaluations, we follow the pessimistic `null` comparison semantics. That means, we define $\perp = \perp$ and $\perp \neq x$ for any (admissible) value x. This is in line with related work on data profiling, see [AN11; AQN14; Hei+13; PN17; Sis+06]. More detailed interpretations of `null` markers for UCCs use so-called possible world and certain world models [Köh+16], leading to specialized definitions and discovery approaches [WLL19]. Research on null semantics is not our focus, hence we resort to the practical definition. We note that all algorithms presented in this thesis can easily be adapted to cover all standard `null` semantics.

2.3.3 Functional and Inclusion Dependencies

Functional dependencies (FDs) over a schema R are expressions of the form $X \to a$ for some set $X \subseteq R$ and a single attribute $a \in R$. The set X is the *left-hand side* (LHS) of the dependency and a is the *right-hand side* (RHS). We say that the FD has *size* $|X|$. An FD $X \to a$ *holds*, or is *valid*, in a database $\mathfrak{r}$ (over R) if any pair of records that agree on X also agree on a, that is, if $r[X] = s[X]$ implies $r[a] = s[a]$ for any $r, s \in \mathfrak{r}$. Otherwise, $X \to a$ is said to *fail* in $\mathfrak{r}$, or be *invalid*. Intuitively, $X \to a$ holds if the value of attribute a is a functions of the values appearing in X. The FD $\emptyset \to a$ holds iff all rows agree on a. A *generalization* of an FD $X \to a$ is another FD $X' \to a$ that is valid and satisfies $X' \subsetneq X$. An FD $X \to a$ is *(inclusion-wise) minimal* if it holds in $\mathfrak{r}$ and does not have a generalization, that is, if $X' \to a$ fails for any proper subset $X' \subsetneq X$. A functional dependency is *non-trivial* if $a \notin X$. Observe that trivial FDs hold in any database.

Inclusion dependencies model the case where values appearing in one database are also contained in another, potentially over a different schema. Let R and S be two relational schemas and $\mathfrak{r}$ and $\mathfrak{s}$ databases over R and S, respectively. For some $X \subseteq R$, let $\sigma\colon X \to S$ be an injective map. The pair (X, σ) is an *inclusion dependency* (IND) if, for each row $r \in \mathfrak{r}$, there exists some $s \in \mathfrak{s}$ such that $r[a] = s[\sigma(a)]$ for every $a \in X$, that is, $\mathfrak{r}[X] \subseteq \mathfrak{s}[\sigma(X)]$.

If the map σ is given in the input, we say that X is the dependency. A *generalization* of X is another inclusion dependency with $X' \supsetneq X$. Note that here the subset relation is in the opposite direction compared to UCCs and FDs. An inclusion dependency then is *maximal* if the set X is inclusion-wise maximal among all INDs with map σ between $\mathfrak{r}$ and $\mathfrak{s}$. In the general case with arbitrary mappings, we define a partial order on the pairs (X, σ). We say that $(X, \sigma) \preccurlyeq (X', \sigma')$ holds if $X \subseteq X'$ and σ is the restriction of σ' to X. An inclusion dependency then is *maximal* if it is an $\preccurlyeq$-maximal element among the inclusion dependencies between $\mathfrak{r}$ and $\mathfrak{s}$. The notion of a *generalization* is adjusted accordingly. INDs are indeed downward closed with respect to $\preccurlyeq$. However, it may happen that (X, σ) and (X', σ') are both maximal although $X' \subsetneq X$ is a proper subset.

2.4 Computational Complexity

We are mainly developing and analyzing enumeration algorithms. Computational decision problems are a means for this purpose. For the latter, we adopt

a parameterized view and additionally borrow some ideas from fine-grained complexity. We only review the more specific notions here.

2.4.1 Enumeration Complexity

Enumeration is the task of compiling and outputting a list of all solutions to a computational problem without repetitions. Note that this is different from a counting problem, which asks for the mere number of solutions. More formally, an *enumeration problem* is a function $\Pi\colon \{0,1\}^* \to \mathcal{P}(\{0,1\}^*)$ such that, for all *instances* $I \in \{0,1\}^*$, the set of *solutions* $\Pi(I)$ is finite. An algorithm solving this problem needs to output, on input I, all elements of $\Pi(I)$ exactly once, see [CS19]. Unless explicitly stated otherwise, we do not impose any order on the output. We focus on the enumeration of minimal hitting sets, that is, $\Pi\colon (V, \mathcal{H}) \mapsto \mathrm{Tr}(\mathcal{H})$, and minimal unique column combinations, $\Pi'\colon (R, \mathfrak{r}) \mapsto \mathrm{Tr}(\mathcal{D})$, in the notation of the previous section.

The former is known as the *transversal hypergraph problem*. There exists a class of hypergraphs such that the number $N_{\min}$ of solutions growths exponentially in both the number of vertices $n = |V|$ and the number of edges $m = |\mathcal{H}|$. As an example, let $\mathcal{H}$ be a matching on $2k$ vertices for some positive integer k, that is, $\mathcal{H} = \{u_i, v_i\}_{1 \le i \le k}$. Then, $\mathcal{H}$ has 2^k minimal transversals, namely, all sets $\{x_1, \ldots, x_k\}$ with $x_i = u_i$ or $x_i = v_i$. This rules out any polynomial algorithm for the enumeration problem. Instead, one could ask for an *output-polynomial*[5] algorithm running in time polynomial in the combined input and output size, such an algorithm terminates within $\mathbf{poly}(n, m, N_{\min})$ steps.

A (seemingly) stronger requirement is an *incremental polynomial* algorithm, generating the solutions in such a way that the i-th *delay*, the time between the $(i-1)$-st and i-th output, is bounded by $\mathbf{poly}(n, m, i)$. This includes the preprocessing time until the first solution arrives ($i = 1$) as well as the postprocessing time between the last solution and termination ($i = N_{\min} + 1$). We note that there is also an alternative definition of incremental polynomial time which requires that there are polynomials $p, q\colon \mathbb{N} \to \mathbb{N}$ such that, for every positive integer $i \le N_{\min}$, the first i solutions are enumerated in time $p(i)\, q(n+m)$. Both notions are equivalent if the algorithms is allowed to take exponential space in n and m [CS19]. Throughout this thesis, we exclusively use the former definition.

5 Output-polynomial algorithms are sometimes said to run in *polynomial total time* [Str13], we avoid this term due to ambiguities.

The strongest form of output-efficiency is that of *polynomial delay*, where the delay is universally bounded by $\mathbf{poly}(n, m)$. Besides the execution time, one can also restrict the memory consumption of an enumeration algorithm. Ideally, it uses space polynomial in the input size only.

In the particular case of enumerating minimal hitting sets, it is known that there exists an output-polynomial algorithm if and only if there is an incremental polynomial one [BI95]. Polynomial delay seems to be a strictly stronger notion. It is the major open question in the field whether the transversal hypergraph problem can actually be solved in output-polynomial time.

Arguably the most restrictive way to relate enumeration problems are *parsimonious reductions*.[6] Such a reduction from problem Π to Π' is a pair of polynomial time computable functions $f\colon \{0,1\}^* \to \{0,1\}^*$ and $g\colon (\{0,1\}^*)^2 \to \{0,1\}^*$ such that, for any instance $I \in \{0,1\}^*$, $g(I,\cdot)$ is a bijection from $\Pi'(f(I))$ onto $\Pi(I)$. The behavior of $g(I,\cdot)$ on $\{0,1\}^* \setminus \Pi'(f(I))$ is irrelevant. Intuitively, any enumeration algorithm for Π' can then be turned into one for Π by first mapping the input I to $f(I)$ and translating the output solutions back via g. Note that Observation 2.2 establishes a parsimonious reduction from the enumeration of minimal UCCs to the transversal hypergraph problem with $f\colon (R, \mathfrak{r}) \mapsto (R, \mathcal{D})$ and $g((R, \mathfrak{r}), \cdot)$ being the identity over subsets of R.

2.4.2 Parameterized Complexity

The central idea of *parameterized complexity* is to identify a quantity of the input to a decision problem, other than its size, that captures the computational hardness. See the textbooks [Cyg+15; DF13; FG06; Nie06] for a more thorough introduction. For an instance $I \in \{0,1\}^*$, let $|I|$ denote some expressive quantification of the input size of I, polynomially related to the bit length. The *decision problem* associated with a language $\Pi \subseteq \{0,1\}^*$ is to determine whether $I \in \Pi$. A decision problem is *parameterized* if any instance I is additionally augmented with a *parameter* $k = k(I) \in \mathbb{N}^+$, we thus have $\Pi \subseteq \{0,1\}^* \times \mathbb{N}^+$. A parameterized decision problem Π is *fixed-parameter tractable* (FPT), if there exists a computable function $f\colon \mathbb{N}^+ \to \mathbb{N}^+$ and an algorithm that decides any instance (I, k) in time $f(k) \cdot \mathbf{poly}(|I|)$. Intuitively, for FPT-problems, a family of instances for which the parameter is bounded can be solved in polynomial time and the

6 The concept of parsimonious reductions between enumeration problems is inspired by, but should not be confused with, the homonymous reductions for counting problems [CS19].

degree of the polynomial is independent of the parameter. The complexity class **FPT** is the collection of all fixed-parameter tractable problems.

Let Π and Π' be two parameterized problems. A *parameterized reduction*, or *FPT-reduction*, from Π to Π' is an algorithm running in time $f(k) \cdot \mathbf{poly}(|I|)$ on instances (I, k), which outputs some instance (I', k') such that $k' \leq g(k)$ holds for some computable function $g\colon \mathbb{N}^+ \to \mathbb{N}^+$, and $(I, k) \in \Pi$ is true if and only if $(I', k') \in \Pi'$ is. Due to the time bound, we have $|I'| \leq f(k)\,\mathbf{poly}(|I|)$.[7] Thus, any (hypothetical) FPT-algorithm for Π' yields an FPT-algorithm for Π. If additionally there is an FPT-reduction also from Π' to Π, we say that the problems are *FPT-equivalent*. A notable special case of FPT-reductions are *linear* parameterized reductions in which g is a linear function, that is, $k' = \mathrm{O}(k)$. Even more restrictive are polynomial many-one reductions that *preserve* the parameter, meaning $k' = k$.

Parameterized reductions give rise to the **W**-*hierarchy* of complexity classes by specifying a complete problem for each class. We discuss two slightly different families of such problems, one each on Boolean circuits and formulas. A *(Boolean) circuit* is a directed acyclic graph whose vertex set consists of input nodes, NOT-, AND- and OR-gates with the obvious semantics, and a single output node. AND- and OR-gates have potentially unbounded fan-in. The *depth* of a circuit is the maximum length of a directed path from an input to the output node. The *weft* is the maximum number of so-called *large gates* with fan-in greater than 2 on any path. The Weighted Circuit Satisfiability problem is to decide for a given circuit and a positive integer k whether the circuit has a satisfying assignment of *(Hamming) weight* k, that is, with exactly k input nodes set to true. The budget k serves as the parameter.

For every positive integer t, $\mathbf{W}[t]$ is defined to be the class of all parameterized problems that admit an FPT-reduction to Weighted Circuit Satisfiability restricted to circuits of constant depth and weft at most t. When taking all parameterized problems that allow an FPT-reduction to the unrestricted Weighted Circuit Satisfiability problem, we get $\mathbf{W}[\mathbf{P}]$.

Before introducing the second family, consider the Independent Set problem on graphs[8] parameterized by the size of the sought solution. This problem is com-

7 This is intended to mean that there exists a non-negative integer ℓ such that $|I'| \in \mathrm{O}(f(k) \cdot |I|^{\ell})$. We find the chosen notation to be more suggestive.

8 The Independent Set problem is to decide for a graph and a positive integer k whether the graph has a set of k vertices such that no two of them share an edge.

plete for the class $\mathbf{W}[1]$. A *(Boolean) formula* is a circuit in which every logic gate has fan-out 1. The input nodes, the *variables*, may have unbounded fan-out. For some positive integer t, a formula is *t-normalized* if it is a conjunction of disjunctions of conjunctions of disjunctions (and so on) of literals with $t-1$ alternations or, equivalently, t levels of AND- and OR-gates starting with an AND-level at the output node. The Weighted Circuit Satisfiability problem on t-normalized formulas is called Weighted t-normalized Satisfiability. For every $t \geq 2$, Weighted t-normalized Satisfiability is $\mathbf{W}[t]$-complete.[9] The class $\mathbf{W}[\mathrm{SAT}]$ is the class of all parameterized problems admitting an FPT-reduction to Weighted Circuit Satisfiability on arbitrary formulas.

The classes $\mathbf{FPT} \subseteq \mathbf{W}[1] \subseteq \mathbf{W}[2] \subseteq \cdots \subseteq \mathbf{W}[\mathrm{SAT}] \subseteq \mathbf{W}[\mathbf{P}]$ form an ascending hierarchy. All inclusion are conjectured to be strict, which is, however, unproven. The whole $\mathbf{W}$-hierarchy is contained in the class (uniform) $\mathbf{XP}$ of all parameterized problems whose instances (I, k) can be decided in time $|I|^{f(k)}$ for computable f.

2.4.3 Fine-Grained Complexity

Fine-grained complexity aims to determine the exact exponent of the best worst-case running time of any algorithm for some decision problem. The more precise analysis comes at the cost of results often relying on unproven hypotheses. The first two such conjectures treat the satisfiability of Boolean formulas. For a positive integer k, the k-CNF Satisfiability problem is to decide the existence of a satisfying assignment to the variables of a formula in conjunctive normal form (CNF) where each clauses has exactly k literals.

▶ **Hypothesis 2.3 (Strong Exponential Time Hypothesis, SETH [IP01; IPZ01]).** For every constant $\varepsilon > 0$, there exists a positive integer $k = k(\varepsilon)$ such that the k-CNF Satisfiability problem on formulas with n variables does not admit an algorithm running in time $O(2^{(1-\varepsilon)n})$. ◀

▶ **Hypothesis 2.4 (Exponential Time Hypothesis, ETH [IP01; IPZ01]).** There exists a constant $\varepsilon > 0$, such that the 3-CNF Satisfiability problem on formulas with n variables does not admit an algorithm running in time $O(2^{\varepsilon n})$. ◀

9 The definition via normalized formulas comes with an inconsistency at $t = 1$. A 1-normalized formula is a single conjunctive clause, the associated weighted satisfiability problem is trivially seen to be in $\mathbf{P}$ and thus in $\mathbf{FPT}$.

SETH indeed implies ETH [IP01], an inverse connection is not known. So-called fine-grained reductions[10] transfer the conjectured hardness of k-CNF Satisfiability to other problems. We use two results of this kind pertaining to problems in **P** and **W**[1], respectively.

The Orthogonal Vectors problem is to decide, given two sets A and B of n binary vectors each in d dimensions, whether there are $\vec{a} \in A$ and $\vec{b} \in B$ such that $\vec{a} \cdot \vec{b} = \sum_{i=1}^{d} a_i b_i = 0$. The problem has a trivial $O(n^2 d)$-time algorithm and the Orthogonal Vectors Conjecture claims that this is essentially optimal, up to sub-polynomial factors. There are several different formulations of this hypothesis in the literature, we follow the nomenclature by Gao et al. [Gao+18].

▶ **Hypothesis 2.5 (Orthogonal Vectors conjecture in moderate dimensions (OV conjecture)).** For every constant $\varepsilon > 0$, the Orthogonal Vectors problem does not admit an algorithm running in time $O(n^{2-\varepsilon}) \cdot \mathsf{poly}(d)$. ◀

The OV conjecture is lent credibility by the fact that it is implied by SETH [Wil05], any truly subquadratic algorithm for Orthogonal Vectors would thus lead to a major breakthrough in satisfiability. Chen et al. [Che+06] showed a similar connection between ETH and parameterized complexity.

▶ **Proposition 2.6 (Chen et al. [Che+06]).** Assume ETH holds. Then, the Independent Set problem parameterized by the solution size k on n-vertex and m-edge graphs does not admit an algorithm running in time $f(k) \cdot (n+m)^{o(k)}$ for any computable function f. Moreover, if there is a linear parameterized reductions from Independent Set, to some parameterized problem Π', then Π' does not admit an algorithm running in time $f(k') \cdot |I'|^{o(k')}$. ◀

The proposition states that any worst-case running time for Independent Set has a linear dependency on k in the exponent. Since the problem is in **W**[1], ETH implies **W**[1] ≠ **FPT**. The second part of the assertion can easily be extended to the case in which the parameterized reduction to Π' is not linear, but observes $k' = g(k)$ for some bijective computable function g. In this case, Π' cannot be solved in time $f(k') \cdot |I'|^{o(g^{-1}(k'))}$, unless ETH fails.

10 We are not concerned with the technical details of fine-grained reductions, we refer the interested reader to the work of Vassilevska Williams and Williams [VW18].

2.5 Information Theory

Some concepts in probability, and even combinatorics, allow for a very concise and elegant description using the language of information theory. In what follows, we intend the expressions $0 \log_a 0$ and $0 \log_a(\frac{0}{0})$ to both mean 0 for any positive real base a. Note that this convention implies $0^0 = a^{0 \log_a 0} = 1$ and $(\frac{0}{0})^0 = 1$. We use $\operatorname{ld} x$ for the binary (base-2) logarithm of some positive real x.

Let $\vec{p} = (p_i)_{i \in \mathbb{N}}$ be a discrete probability distribution. Its *entropy*[11] is $H(\vec{p}) = -\sum_{i \in \mathbb{N}} p_i \operatorname{ld} p_i$. If $\vec{p}$ has a finite support, $H(\vec{p})$ can be interpreted as the minimum average word length of any binary code to communicate the outcomes of a random variable $X \sim \vec{p}$. The *entropy function* H is the entropy of the Bernoulli distribution. That is, we have, for any probability x,

$$\mathrm{H}(x) = H((x, 1-x)) = -x \operatorname{ld} x - (1-x) \operatorname{ld}(1-x).$$

The entropy function is symmetric around $1/2$ with $\mathrm{H}(x) = \mathrm{H}(1-x)$. On the open unit interval, H is positive and strictly concave It has its maximum at $1/2$ with a value of $\mathrm{H}(1/2) = 1$. The entropy power $2^{\mathrm{H}(x)} = 1/(x^x (1-x)^{1-x})$ is the *perplexity* of x. We use it to estimate binomial coefficients, see [CT06].

▶ **Proposition 2.7.** Let n be a positive integer and $0 < x < 1$ a rational number such that xn is an integer, then

$$\frac{2^{\mathrm{H}(x)n}}{\sqrt{8nx(1-x)}} \leq \binom{n}{xn} \leq \frac{2^{\mathrm{H}(x)n}}{\sqrt{\pi nx(1-x)}}. \qquad \blacktriangleleft$$

Let $\vec{q} = (q_i)_{i \in \mathbb{N}}$ be a second distribution such that $\vec{p}$ is absolutely continuous with respect to $\vec{q}$, meaning that, for all i, $q_i = 0$ implies $p_i = 0$. The *divergence*[12] to $\vec{p}$ from $\vec{q}$ is $D(\vec{p} \,\|\, \vec{q}) = -\sum_i p_i \operatorname{ld}(\frac{q_i}{p_i})$. Note that in general the divergence is not symmetric, that is, $D(\vec{p} \,\|\, \vec{q}) \neq D(\vec{q} \,\|\, \vec{p})$. In terms of binary codes, $D(\vec{p} \,\|\, \vec{q})$ is the average number of bits wasted when communicating the outcomes of $X \sim \vec{p}$ with a code that is optimized for the distribution $\vec{q}$.

11 In full detail, this is the *binary Shannon entropy* or, equivalently, the *binary Rényi entropy of order* 1. We do not use any other kind of entropy in this thesis.

12 In full detail, this is the *binary Kullback–Leibler divergence* or, equivalently, the *binary Rényi divergence of order* 1. It is sometimes also called *relative entropy* [CT06], we avoid this term due to ambiguities. We do not use any other kind of divergence in this thesis.

We are only interested in the divergence between Bernoulli distributions. Let x and y be two probabilities and define

$$\mathrm{D}(x \,\|\, y) = D((x, 1-x) \,\|\, (y, 1-y)) = -x \operatorname{ld}\left(\frac{y}{x}\right) - (1-x) \operatorname{ld}\left(\frac{1-y}{1-x}\right).$$

The function D is convex in both x and y, and attains its minimum 0 for $x = y$. It observes $\mathrm{D}(x \,\|\, y) = \mathrm{D}(1-x \,\|\, 1-y)$ and its partial derivative with respect to x is $\frac{\partial}{\partial x} \mathrm{D}(x \,\|\, y) = \operatorname{ld}\left(\frac{x}{1-x} \frac{1-y}{y}\right)$.

We show next that the divergence scales quadratically in the difference $y - x$.

▶ **Lemma 2.8.** Let $0 < x \le y < 1$ be two non-trivial probabilities. Define t^+ to be the maximizer of $t(1-t)$ over the interval $[x, y]$, and t^- the minimizer. Then,

$$\frac{(y-x)^2}{t^+(1-t^+)} \le 2 \ln(2) \cdot \mathrm{D}(x \,\|\, y) \le \frac{(y-x)^2}{t^-(1-t^-)}. \qquad \blacktriangleleft$$

Proof. Let $\varepsilon = y - x$. The function $\mathrm{D}(y - \varepsilon \,\|\, y)$ is two-times differentiable w.r.t. ε,

$$\frac{\partial}{\partial \varepsilon} \mathrm{D}(y-\varepsilon \,\|\, y) = \operatorname{ld}\left(\frac{y}{1-y} \frac{1-y+\varepsilon}{y-\varepsilon}\right); \quad \frac{\partial^2}{\partial \varepsilon^2} \mathrm{D}(y-\varepsilon \,\|\, y) = \frac{1}{\ln 2} \frac{1}{(y-\varepsilon)(1-y+\varepsilon)}.$$

The divergence and its first derivative vanish at $\varepsilon = 0$. By Taylor's theorem, there exists a real number ξ with $0 \le \xi \le \varepsilon$ such that

$$\mathrm{D}(y - \varepsilon \,\|\, y) = \frac{\varepsilon^2}{2!} \cdot \frac{\partial^2}{\partial \varepsilon^2} \mathrm{D}(y - \varepsilon \,\|\, y)\Big|_{\varepsilon=\xi} = \frac{\varepsilon^2}{2 \ln 2 \,(y-\xi)(1-y+\xi)}.$$

The lemma follows from $y - \xi$ ranging over $[x, y]$. ■

We mainly use the following divergence power, which closely resembles the perplexity, $2^{-\mathrm{D}(x \,\|\, y)} = 2^{\mathrm{H}(x)} \cdot y^x (1-y)^{1-x} = \left(\frac{y}{x}\right)^x \left(\frac{1-y}{1-x}\right)^{1-x}$. The next lemma relates those quantities for different parameters.

▶ **Lemma 2.9.** Let $0 \le x \le y \le z \le 1$ be three probabilities, then

$$2^{-\mathrm{D}(x \,\|\, z)} = \left(\frac{y}{1-y} \frac{1-z}{z}\right)^{y-x} \cdot 2^{-\mathrm{D}(x \,\|\, y)} \cdot 2^{-\mathrm{D}(y \,\|\, z)}.$$

In particular, for any fixed z, $2^{-\mathrm{D}(x \,\|\, z)}$ is non-decreasing in x as long as $x \le z$. ◀

Proof. The convention $(\frac{0}{0})^0 = 1$ ensures that $\left(\frac{y}{1-y}\frac{1-z}{z}\right)^{y-x}$ is well-defined.

$$\frac{2^{-\mathrm{D}(x\,\|\,z)}}{2^{-\mathrm{D}(y\,\|\,z)}} = \frac{\left(\frac{z}{x}\right)^x\left(\frac{1-z}{1-x}\right)^{1-x}}{\left(\frac{z}{y}\right)^y\left(\frac{1-z}{1-y}\right)^{1-y}} = \frac{\left(\frac{z}{x}\right)^x\left(\frac{1-z}{1-x}\right)^{1-x}}{\left(\frac{z}{y}\right)^{y-x}\left(\frac{z}{y}\right)^x\left(\frac{1-z}{1-y}\right)^{1-x}\left(\frac{1-z}{1-y}\right)^{x-y}}$$

$$= \frac{1}{\left(\frac{z}{y}\right)^{y-x}\left(\frac{1-z}{1-y}\right)^{x-y}}\left(\frac{y}{x}\right)^x\left(\frac{1-y}{1-x}\right)^{1-x} = \left(\frac{y}{1-y}\,\frac{1-z}{z}\right)^{y-x} 2^{-\mathrm{D}(x\,\|\,y)}.$$

The monotonicity follows from the last two factors being at most 1. ■

2.6 Polynomials of Probabilities

We occasionally need to estimate expressions of the form $(1-x)^n$ where x is a probability. The first inequality for this task is taken from the textbook [MR95].

▶ **Proposition 2.10.** Let n be a positive integer and x a real such that $|x| \le n$,

$$e^x\left(1 - \frac{x^2}{n}\right) \le \left(1 + \frac{x}{n}\right)^n. \qquad \blacktriangleleft$$

We reach rather tight bounds on $(1-x)^n$ by substituting x for $-nx$ above, and combining it with the simple fact that $(1+x) \le e^x$ holds for all x.

▶ **Corollary 2.11.** Let n be a non-negative integer and x a probability, then

$$\left(1 - nx^2\right) \le e^{nx}(1-x)^n \le 1. \qquad \blacktriangleleft$$

The last inequality was given by Badkobeh, Lehre, and Sudholt [BLS15].

▶ **Proposition 2.12 (Lemma 10 in [BLS15]).** Let n be a non-negative integer and x a probability, then

$$\frac{nx}{1+nx} \le 1 - (1-x)^n \le nx. \qquad \blacktriangleleft$$

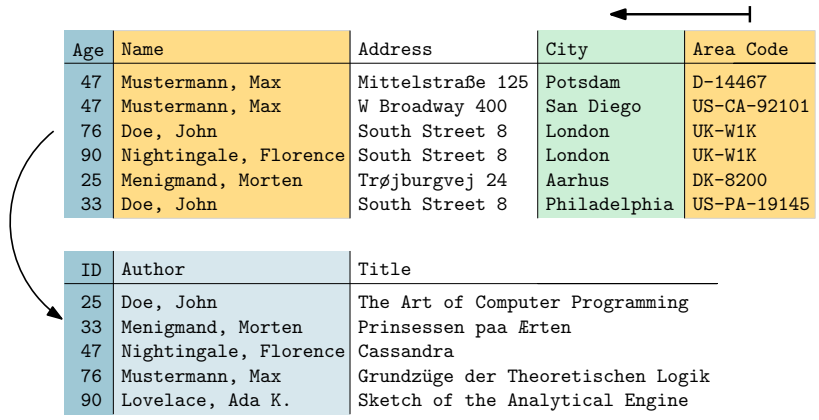
3 Complexity of Dependency Detection and Discovery

We prove the parameterized complexity of deciding the existence of certain multi-column dependencies in relational databases with the solution size as parameter. We prove that detecting unique column combinations and functional dependencies is **W**[2]*-complete, while the corresponding problem for inclusion dependencies is* **W**[3]*-complete. We then use those results to settle the enumeration complexity of the associated discovery problems under parsimonious reductions.*

3.1 Introduction

The ability to discover multi-column dependencies from unannotated relational databases is paramount for data profiling. We have already seen the importance of unique column combinations as fingerprints of the data that allow subsequent query optimizations. Another type of dependencies are the *functional dependencies* (FDs). They model the case in which we are only interested in identifying the values of a specific column, instead of all columns. In the extended example in Figure 3.1, if one knows the `Name` and `Area Code` one can infer all other values since the columns form a UCC. If one is only interested in the `City`, however, it is enough to use the `Area Code` since it already determines the latter, the `Name` is not needed. More formally, the FD `Area Code` $\rightarrow$ `City` is valid and minimal.

We also investigate *inclusion dependencies* (INDs) that reveal connections between *different* databases. A unary IND holds if all values in a column of the first database are also contained in one column of the second database. All values of the `Name` attribute in the first table of Figure 3.1 appear again as an `Author` in the second table. The same holds, maybe accidentally, for the `Age` and `ID`. An inclusion dependency has higher arity if the inclusion also pertains to the tuples of values in multiple columns. Note that this is not the case in Figure 3.1. In contrast to UCCs and FDs–where we want solutions to be small–we ought to find large and ideally maximal inclusion dependencies. Those are much more likely to be caused by the inherent structure of the data than by mere coincidence.



Age	Name	Address	City	Area Code
47	Mustermann, Max	Mittelstraße 125	Potsdam	D-14467
47	Mustermann, Max	W Broadway 400	San Diego	US-CA-92101
76	Doe, John	South Street 8	London	UK-W1K
90	Nightingale, Florence	South Street 8	London	UK-W1K
25	Menigmand, Morten	Trøjburgvej 24	Aarhus	DK-8200
33	Doe, John	South Street 8	Philadelphia	US-PA-19145

ID	Author	Title
25	Doe, John	The Art of Computer Programming
33	Menigmand, Morten	Prinsessen paa Ærten
47	Nightingale, Florence	Cassandra
76	Mustermann, Max	Grundzüge der Theoretischen Logik
90	Lovelace, Ada K.	Sketch of the Analytical Engine

Figure 3.1: Illustration of multi-column dependencies. The `Name` and `Area Code` together are a minimal UCC in the first database. Due to the uniqueness, the FD `Name, Area Code → City` holds. Its left-hand side is not minimal since the generalization `Area Code → City` is also valid. There are two maximal INDs of size 1 between the first and the second database, `Age` is included in `ID` and `Name` in `Author`. They cannot be combined to an IND of size 2 because, for example, the value combination (33, "`Doe, John`") does not appear in the column combination `ID, Author`.

Similar to UCCs, discovering the functional dependencies and inclusion dependencies of a database (respectively of pairs thereof) is an important preprocessing step intended to improve the subsequent data access. The valid FDs and INDs are used for example in cardinality estimation in query plan optimizers [Ily+04], query rewriting [Gia+02], and joins [CFP84]. For a detailed exposition of the applications of data dependencies see [KPN21].

The *detection* (decision) problems for all three types of dependencies are **NP**-complete.[1] Notwithstanding, detection algorithms often perform well on practical datasets [Abe+18]. One approach to bridge this apparent gap is to analyze whether properties that are usually observed in realistic data benevolently influence the hardness of the problem. Exploiting those properties may even lead to algorithms that guarantee a polynomial running time in case these features are present in the problem instance. This is formalized in the concept of parameterized algorithms [Cyg+15; DF13; Nie06]. The algorithm designers

1 See Sections 3.2.1 and 3.3.1 for precise statements and references.

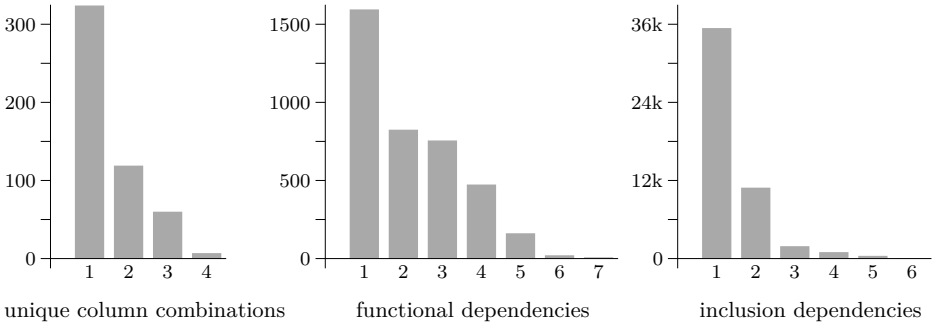


Figure 3.2: The number of minimal unique column combinations, minimal functional dependencies, and maximal inclusion dependencies for varying solution sizes in the `MusicBrainz` database.[2]

then have to identify suitable parameters that can be exploited algorithmically and are actually small in practice.

Consider, for instance, the histograms in Figure 3.2, showing the size distribution of minimal unique column combinations and functional dependencies, as well as maximal inclusion dependencies in the `MusicBrainz` database [Swa02]. The majority of dependencies, of all types, are very small. Beside surrogate keys that give rise to multiple functional dependencies of size 1, causalities in the data can also lead to small FDs. For example, the name of an event together with the year in which it started determines the year in which it ends, implying an FD of size 2. Note that the starting year alone is usually not enough to infer this information. The name of the action, however, seems to indicate whether the event ends in the same year or the next. The size of the dependency is thus a natural candidate for an algorithmic parameter. Notwithstanding, we show that it is unlikely to be the sole explanation for the good practical performance. We prove that the detection of unique column combinations and functional dependencies is $\mathbf{W}[2]$-complete with respect to the size of the sought solution, detecting inclusion dependencies is even $\mathbf{W}[3]$-complete. For all we know, this excludes any algorithm parameterized by the size.

The hardness of detecting INDs is surprising also from a complexity-theoretic standpoint. Currently, there are only a handful of natural problems known to

2 We are thankful to Sebastian Kruse, Felix Naumann, and Thorsten Papenbrock for providing the data presented in the Figure 3.2.

be complete for the class **W**[3]. We show here that the detection of inclusion dependencies has this property. Building on the work presented in Section 3.3, Hannula, Song, and Link [HSL21] have identified independence detection in relational databases as another representative of this class.

We use the insights gained on the detection of multi-column dependencies to also investigate their *discovery* (enumeration). The case of unique column combinations is inseparably related to the *transversal hypergraph* problem, where we are tasked to compute all minimal hitting sets of a given hypergraph, see Observation 2.2. Namely, the minimal UCCs can be discovered in output-polynomial time if and only if the transversal hypergraph problem has an output-polynomial solution [EG95]. Already at its conception, the transversal hypergraph problem was linked to databases [DT87; MR87], but it also emerges in many other applications in fields as diverse as artificial intelligence [Rei87], machine learning [DMP99], distributed systems [GB85], integer linear programming [Bor+02], monotone logic [EMG08], and bioinformatics [Pus+20]. Despite the large interest, the exact complexity of the enumeration problem is still open. In particular, there is no output-polynomial algorithm known. However, many methods work well on practical instances [GV17]. Data profiling is no exception as modern algorithms succeed within reasonable time frames on many real-world databases [Abe+18].

The mentioned equivalence between the discovery of UCCs and the transversal hypergraph problem was proven via a reduction that continuously calls a decision subroutine to check whether the enumeration has found all solutions [EG95]. The construction inherently requires space proportional to the output size and is therefore hardly useful in practice. We are able to radically simplify and at the same time generalize this equivalence by relating unique column combinations, functional dependencies, and hitting sets directly at the enumeration level using *parsimonious reductions*, running in polynomial time and space. We give similar results also for the discovery of maximal inclusion dependencies. The (even closer) connection to the transversal hypergraph problem explains in parts why dependency discovery works quite well on real-world databases. Moreover, it allows us to transfer ideas from the design of hitting set enumeration algorithms to data profiling, thereby connecting the two research areas. For example, there are very space-efficient algorithms known for the transversal hypergraph problem, while memory consumption still seems to be a major obstacle in dependency discovery [PN16; WLL19]. Indeed, the foundations laid here will later lead to improved profiling algorithms in Chapters 4 and 7.

Main Results. We settle the parameterized complexity of the cardinality-constrained decision problems for unique column combinations, functional dependencies, and inclusion dependencies in relational databases, with the solution size as parameter. We prove the following theorems.

▶ **Theorem 3.1.** Detecting a unique column combination of size k in a relational database is $\mathbf{W}[2]$-complete when parameterized by k. The same is true for the detection of a valid, non-trivial functional dependency with a left-hand side of size at most k, even if the desired right-hand side is given in the input. ◀

▶ **Theorem 3.2.** Detecting an inclusion dependency of size k in a pair of relational databases is $\mathbf{W}[3]$-complete when parameterized by k. The result remains true even if both databases are over the same relational schema with the identity mapping between their columns. ◀

We also characterize the complexity of enumerating all multi-column dependencies of a certain type in a database. We do so by proving parsimonious equivalences with well-known enumeration problems and generalizations thereof.

▶ **Theorem 3.3.** The following enumeration problems are equivalent under parsimonious reductions:

(*i*) discovering the minimal unique column combinations of a relational database;

(*ii*) discovering the minimal, valid, and non-trivial functional dependencies of a relational database with a fixed right-hand side;

(*iii*) the transversal hypergraph problem.

The discovery of functional dependencies with arbitrary right-hand sides is equivalent to simultaneously enumerating the hitting sets of multiple input hypergraphs. The latter two problems are at least as hard as the transversal hypergraph problem. ◀

▶ **Theorem 3.4.** The following enumeration problems are equivalent under parsimonious reductions:

(*i*) discovering the maximal inclusion dependencies of a pair of relational databases;

(*ii*) enumerating the maximal satisfying assignments of an antimonotone, 3-normalized Boolean formula.

This remains true even if the two databases are over the same schema and only inclusions between the same columns are allowed. All those problems are at least as hard as the transversal hypergraph problem. ◀

We also briefly discuss the consequences of our findings to the approximability of minimum dependencies.

3.2 Unique Column Combinations and Functional Dependencies

Theoreticians as well as practitioners in data profiling and database design are frequently confronted with the task of finding a small collection of items that has a non-empty intersection with each member of a prescribed family of sets, see [Abe+18; Dat03; DR94; Kan+92; Mai83]. They thus aim to solve instances of the hitting set problem. In this section, we show that this encounter is inevitable in the sense that detecting a single small unique column combination or functional dependency in a relational database is the same as finding a hitting set in a hypergraph. Even more so, this equivalence extends to enumeration. We show that the associated discovery problems of finding all UCCs or FDs is indeed the same as enumerating all hitting sets.

We first formally define the respective decision and enumeration problems we discuss here. The decision versions are always parameterized by the solution size. We then order them in a (seemingly ascending) chain via parameterized reductions. However, the beginning and end of this chain will turn out to be FPT-equivalent, which settles the complexity of the problems involved as complete for the parameterized complexity class $\mathbf{W}[2]$. We then use our gained insights to also show the equivalence of the corresponding enumeration problems.

3.2.1 Problem Definition

Recall the definitions of hitting sets as well as unique column combinations and functional dependencies from Sections 2.2 and 2.3. We are interested in the parameterized complexity of the associated cardinality-constrained decision problems. The constraint always serves as the parameter.

Hitting Set

Instance: A hypergraph $(V, \mathcal{H})$ and a non-negative integer k.

Parameter: The non-negative integer k.

Decision: Is there a set $T \subseteq V$ with $|T| = k$ such that T is a hitting set for $\mathcal{H}$?

Note that if $k > |V|$, then the answer to the decision problem is trivially FALSE; otherwise, there is no difference between deciding the existence of a transversal with *at most* or *exactly* k elements since every superset of a hitting set is again a hitting set. We ignore the special case of a too large k as parameterized complexity is primarily concerned with the situation that the parameter is much smaller than the input size. The unparameterized Hitting Set problem is one of Karp's initial 21 **NP**-complete problems [Kar72]. In fact, its minimization variant is even **NP**-hard to approximate[3] within a factor of $(1 - \varepsilon) \ln |V|$ for any constant $\varepsilon > 0$ [DS14]. The parameterized decision problem is **W**[2]-complete [DF13].

The corresponding enumeration problem broadens the notion of a "small" solution, the task is now to list all *inclusion-wise minimal* hitting sets, that is, the edges of the transversal hypergraph $\mathrm{Tr}(\mathcal{H})$. All other (non-minimal) hitting sets can be trivially obtained from the minimal ones by arbitrarily adding more vertices. The term *enumeration* contains the requirement not to repeat outputs.

Transversal Hypergraph

Instance: A hypergraph $(V, \mathcal{H})$.

Enumeration: List all edges of $\mathrm{Tr}(\mathcal{H})$.

Let $N = |\mathcal{H}| + |\mathrm{Tr}(\mathcal{H})| + |V|$ denote the combined input and output size, the fastest known algorithm to enumerate[4] $\mathrm{Tr}(\mathcal{H})$ runs in time $N^{\mathrm{O}(\log N / \log\log N)}$ [FK96].

We generalize the problem to the enumeration of minimal hitting sets for multiple input hypergraphs. We do not prescribe any order on the output, in particular the hypergraphs do not need to be processed one by one. However, we want to be able to quickly tell to which input a solution belongs. For this, let $\mathrm{Tr}(\mathcal{H}) \dot\cup \mathrm{Tr}(\mathcal{G})$ denote the disjoint union of the transversal hypergraphs of

3 Dinur and Steurer [DS14] show optimal hardness of approximation for Set Cover, which is, given a hypergraph $(V, \mathcal{H})$, finding a subhypergraph $\mathcal{G} \subseteq \mathcal{H}$ with the minimum number of edges such that $V = \bigcup_{E \in \mathcal{G}}$. The folklore reduction to Hitting Set preserves approximations.

4 The algorithm by Fredman and Khachiyan [FK96] was initially conceived to solve the equivalent problem of *dualizing* monotone Boolean formulas, that is, given a CNF formula without negations, compute the equivalent DNF.

$(V, \mathcal{H})$ and $(W, \mathcal{G})$, with the additional requirement that the union is encoded such that, for any $T \in \mathrm{Tr}(\mathcal{H}) \dot{\cup} \mathrm{Tr}(\mathcal{G})$, the containment $T \in \mathrm{Tr}(\mathcal{H})$ is decidable in time $\mathbf{poly}(|V|, |W|, |\mathcal{H}|, |\mathcal{G}|)$ independently of the sizes $|\mathrm{Tr}(\mathcal{H})|$ and $|\mathrm{Tr}(\mathcal{G})|$.

Transversal Hypergraph Union

Instance: A d-tuple of hypergraphs $(\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_d)$.

Enumeration: List all edges of $\mathrm{Tr}(\mathcal{H}_1) \dot{\cup} \mathrm{Tr}(\mathcal{H}_2) \dot{\cup} \ldots \dot{\cup} \mathrm{Tr}(\mathcal{H}_d)$.

Clearly, Transversal Hypergraph Union is at least as hard as the Transversal Hypergraph problem.

We now define the detection and discovery problems of multi-column dependencies in relational databases, again starting with the cardinality-constraint decision problems.

Unique Column Combination

Instance: A relational database $\mathfrak{r}$ over schema R and a non-negative integer k.

Parameter: The non-negative integer k.

Decision: Is there a set $U \subseteq R$ with $|U| = k$ such that U is a unique column combination in $\mathfrak{r}$?

The minimization variant[5] of Unique Column Combination is **NP**-hard [SR92].

Discover Minimal UCCs

Instance: A relational database $\mathfrak{r}$.

Enumeration: List all minimal unique column combinations of $\mathfrak{r}$.

Regarding functional dependencies, we define two variants of the decision problem that slightly differ in the given input. The first one fixes the right-hand side of the desired dependency, while the second one asks for an FD with arbitrary RHS holding in the database. The parameterized complexity of their detection will turn out to be the same, but there are differences in their discovery.

5 Skowron and Rauszer [SR92] consider the problem of finding a minimum *reduct*, defined in rough set theory. A reduct is the same as a minimal unique column combination in a knowledge representation system (a database), see [Paw91].

Functional Dependency$_{\textit{fixed RHS}}$

Instance: A relational database $\mathfrak{r}$ over schema R, an attribute $a \in R$, and a non-negative integer k.

Parameter: The non-negative integer k.

Decision: Is there a set $X \subseteq R \backslash \{a\}$ with $|X| = k$ such that the functional dependency $X \longrightarrow a$ holds in $\mathfrak{r}$?

Functional Dependency

Instance: A relational database $\mathfrak{r}$ over schema R and a non-negative integer k.

Parameter: The non-negative integer k.

Decision: Is there a set $X \subseteq R$ with $|X| = k$ and an attribute $a \in R \backslash X$ such that the functional dependency $X \longrightarrow a$ holds in $\mathfrak{r}$?

The unparameterized variant[6] of the Functional Dependency$_{\textit{fixed RHS}}$ problem is **NP**-complete even if the number of admissible values in each column of the database is at most 2 [DR94]. It is virtually the same to ask for a functional dependency whose left-hand side is of size *at most* k; unless $k \geq |R|$ since then no non-trivial FD adheres to the (exact) size constraint. Again, we ignore this.

Recall that we say that a valid functional dependency $X \longrightarrow a$ is minimal if its LHS X is inclusion-wise minimal among all $X' \subseteq R$ such that $X' \longrightarrow a$ is valid.

Discover Minimal FDs$_{\textit{fixed RHS}}$

Instance: A relational database $\mathfrak{r}$ over schema R and an attribute $a \in R$.

Enumeration: List all minimal, valid, non-trivial functional dependencies of $\mathfrak{r}$ with right-hand side a.

Discover Minimal FDs

Instance: A relational database $\mathfrak{r}$.

Enumeration: List all minimal, valid, non-trivial functional dependencies of $\mathfrak{r}$.

Regarding the complexity of Discover Minimal UCCs and Discover Minimal FDs, it is known that they can be solved in output-polynomial time (in incremental polynomial time or with polynomial delay, respectively) if and only if this is possible for the Transversal Hypergraph problem [EG95].

6 Functional Dependency$_{\textit{fixed RHS}}$ appears under the name Min-Features in [DR94].

3.2.2 Detection

Next, we prove the parameterized complexity of the detection problems for unique column combinations and functional dependencies.

We remark that the Unique Column Combination and the Functional Dependency$_{\textit{fixed RHS}}$ problem have been studied before from a parameterized perspective, under different names and in different contexts. Therefore, some of the following results already appear in prior literature. Namely, Lemma 3.5 has been obtained independently by Froese et al. [Fro+16][7] as well as Akutsu and Bao [AB96].[8] Cotta and Moscato [CM03] showed the $\mathbf{W}[2]$-completeness of Functional Dependency$_{\textit{fixed RHS}}$ restricted to binary databases using an approach different from ours.[9] Also, some of the techniques that appear in the following proofs were preempted by articles studying the classical (polynomial) complexity of Unique Column Combination and Functional Dependency$_{\textit{fixed RHS}}$, for example by Davies and Russel [DR94]. Our work can be seen as a unifying framework that also incorporates the more general Functional Dependency problem (with arbitrary right-hand side) and proofs the parameterized complexity of the detection problems in a way that additionally sheds some light on the enumeration complexity of the related discovery problems.

We start by proving that Unique Column Combination is hard for $\mathbf{W}[2]$. Recall from Section 2.4.2 that a parameterized reduction is *parameter-preserving* if the parameter of the resulting instance is the same as that of the input.

▶ **Lemma 3.5.** There is a polynomial, parameter-preserving reduction from Hitting Set to the Unique Column Combination problem. ◀

Proof. Let $(V, \mathcal{H})$ be the hypergraph given in the input to the Hitting Set problem. Without loss of generality, we can assume it to be Sperner; otherwise, we replace it by its minimization $\min(\mathcal{H})$ (in quadratic time). Observe that $\min(\mathcal{H})$ has a hitting set of size at most k iff $\mathcal{H}$ has one. We construct from $\mathcal{H}$ in polynomial time a database $\mathfrak{r}$ over schema V such that the minimal difference sets of $\mathfrak{r}$ are the edges of $\mathcal{H}$. The lemma then follows immediately from Observation 2.2. In particular, since Observation 2.2 transfers solutions, the parameter k is preserved by the reduction.

7 The Unique Column Combination problem is called Distinct Vectors in [Fro+16].
8 It is called Minimum Key-II in [AB96].
9 Functional Dependency$_{\textit{fixed RHS}}$ is called k-Feature Set in [CM03].

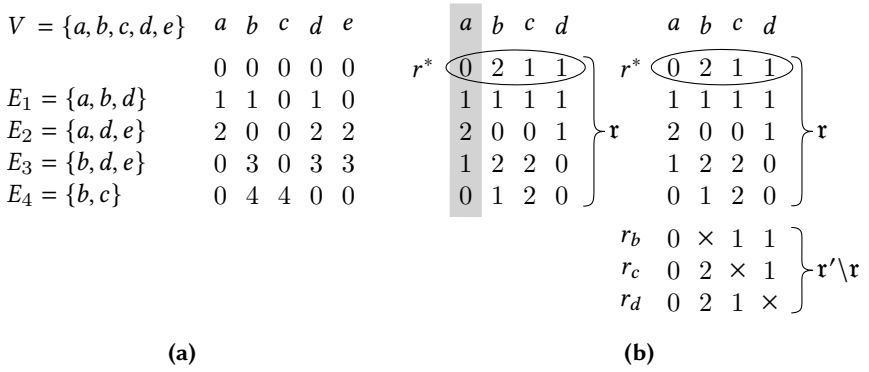


Figure 3.3: Illustration of the Lemmas 3.5 and 3.6. Figure (a) shows an instance of Hitting Set and the equivalent instance of Unique Column Combination. In (b) an instance $\mathfrak{r}$ of Functional Dependency$_{\textit{fixed RHS}}$ with fixed right-hand side a is reduced to instance $\mathfrak{r}'$ of Functional Dependency. The functional dependencies $ab \to d$ holds in $\mathfrak{r}$, but not in $\mathfrak{r}'$.

Let $E_1, E_2, \ldots, E_m$ be the edges of $\mathcal{H}$. We take the integers $0, 1, \ldots, m$ as the admissible values for all attributes in V. As rows, we first add the all-zero tuple $r_0 = (0)_{a \in V}$ to $\mathfrak{r}$. For each $i \in [m]$, we also add the record r_i defined as

$$r_i[a] = \begin{cases} i, & \text{if } a \in E_i; \\ 0, & \text{otherwise.} \end{cases}$$

See Figure 3.3 (a) for an illustration. Clearly, $\mathfrak{r}$ can be computed in linear time.

Any edge E_i is a difference set in $\mathfrak{r}$, namely, that of the pair (r_0, r_i). Any other difference set must come from a pair (r_i, r_j) with $1 \leq i < j \leq m$. It is easy to see that those rows disagree in $E_i \cup E_j$, which is not minimal. Since $\mathcal{H}$ is Sperner, it contains exactly the minimal difference sets of $\mathfrak{r}$. ■

The next two reductions are rather straightforward due to the similar structures of uniques and functional dependencies. While a UCC separates any pair of rows, an FD $X \to a$ needs to distinguish only those with $r[a] \neq s[a]$.

▶ **Lemma 3.6.** There are polynomial, parameter-preserving reductions

(*i*) from Unique Column Combination to Functional Dependency$_{\textit{fixed RHS}}$;

(*ii*) from Functional Dependency$_{\textit{fixed RHS}}$ to Functional Dependency. ◀

Proof. The idea to prove Statement (*i*) is to add a single unique column to the database and fix it as the right-hand side of the sought functional dependency. Let $\mathfrak{r} = \{r_1, r_2, \ldots, r_{|\mathfrak{r}|}\}$ be a database over schema R, and a an attribute not previously in R. We construct $\mathfrak{r}'$ over $R \cup \{a\}$ by adding, for each r_i, the row r'_i defined by $r'_i[R] = r_i[R]$ and $r'_i[a] = i$. The reduction maps an instance $(\mathfrak{r}, R, k)$ of Unique Column Combination to the instance $(\mathfrak{r}', R \cup \{a\}, a, k)$ of Functional Dependency$_{\textit{fixed RHS}}$. Since, for any two distinct rows $r'_i, r'_j \in \mathfrak{r}'$, $i \neq j$, we have $r'_i[a] \neq r'_j[a]$, the left-hand sides of the non-trivial, valid FDs $X \to a$ in $\mathfrak{r}'$ are in one-to-one correspondence to the UCCs in $\mathfrak{r}$.

To reduce Functional Dependency$_{\textit{fixed RHS}}$ to Functional Dependency, we need to mask all "unwanted" FDs with RHS different from the fixed attribute a. See Figure 3.3 (b) for an example. Let again $\mathfrak{r}$ be the input database over R. To construct the resulting database $\mathfrak{r}'$ over the same schema R, we take all rows from $\mathfrak{r}$ and add $|R| - 1$ new ones. Fix an arbitrary record $r^* \in \mathfrak{r}$ and let $\times$ be a new symbol that does not previously appear as a value. For each attribute $b \in R \backslash \{a\}$, we add the row r_b satisfying $r_b[R\backslash\{b\}] = r^*[R\backslash\{b\}]$ and $r_b[b] = \times$. The rows r^* and r_b now witness that any non-trivial FD $X \to b$ fails in $\mathfrak{r}'$.

It is left to prove that $X \to a$ holds in $\mathfrak{r}$ if and only if it holds in $\mathfrak{r}'$. Evidently, any valid FD in $\mathfrak{r}'$ is also valid in the subset $\mathfrak{r}$. Suppose $X \to a$ holds in $\mathfrak{r}$ and let rows $r, s \in \mathfrak{r}'$ be such that $r[a] \neq s[a]$. The only case where the conclusion $r[X] \neq s[X]$ may possibly be in doubt is if $r \in \mathfrak{r}' \backslash \mathfrak{r}$ and $s \in \mathfrak{r}$ (all new rows in $\mathfrak{r}' \backslash \mathfrak{r}$ agree on a). Hence, $r = r_b$ for some $b \neq a$. If $b \in X$, the new value $\times$ appears in the projection $r[X]$ but not in $s[X]$; otherwise, we have $r[X] = r^*[X]$. Since $X \to a$ holds for the pair $r^*, s \in \mathfrak{r}$, the relation $r[X] = r^*[X] \neq s[X]$ follows. ■

The next lemma proves that every instance of the unrestricted Functional Dependency problem can be expressed as an equivalent Boolean formula in conjunctive normal form. Since CNF formulas (of unbounded clause width) are exactly the 2-normalized ones, we obtain a reduction to Weighted 2-normalized Satisfiability. This is the main result of this section.

Cotta and Moscato [CM03] have previously shown the $\mathbf{W}[2]$-completeness of Functional Dependency$_{\textit{fixed RHS}}$. To prove containment in $\mathbf{W}[2]$, they reduce the problem to Weighted Circuit Satisfiability on circuits of weft 2 whose depth is not constant but *a function of the parameter k*, thus formally placing it in what is known as the complexity class $\mathbf{W}^*[2]$. They then apply the equality $\mathbf{W}^*[2] = \mathbf{W}[2]$ [DF98] as a black box. We find it more intuitive to directly reduce

to Weighted 2-normalized Satisfiability. This also seamlessly incorporates the more general version of Functional Dependency for arbitrary RHSs.

▶ **Lemma 3.7.** There is a parameterized reduction from Functional Dependency to the Weighted 2-normalized Satisfiability problem. ◀

Proof. Given a database $\mathfrak{r}$ over R, we derive a CNF formula that has a satisfying assignment of Hamming weight $k+1$ if and only if there is a non-trivial FD with left-hand side of size k that holds in $\mathfrak{r}$. We use two types of variables distinguished by their semantic purpose, $\mathtt{Var}_{LHS} = \{x_a \mid a \in R\}$ and $\mathtt{Var}_{RHS} = \{y_a \mid a \in R\}$. Some variable x_a from $\mathtt{Var}_{LHS}$ being set to true corresponds to the attribute a appearing on the left-hand side of the sought functional dependency; for y_a from $\mathtt{Var}_{RHS}$, this means the attribute is the right-hand side.

We start with the RHS. It might be tempting to enforce that any satisfying assignment chooses exactly one variable from $\mathtt{Var}_{RHS}$. We prove below that this is not necessary and forgoing the $O(|R|^2)$ clauses representing this constraint allows for a (slightly) leaner construction. However, we do have to ensure that at least one variable from $\mathtt{Var}_{RHS}$ is set to true and the corresponding one in $\mathtt{Var}_{LHS}$ is false. This is done by the clauses $C_{RHS} = \bigvee_{y_a \in \mathtt{Var}_{RHS}} y_a$ and $C_a = \neg y_a \vee \neg x_a$ for each $a \in R$. The subformula φ_{RHS} is their conjunction.

We now model the LHS. For any attribute $a \in R$ and rows $r, s \in \mathfrak{r}$, we define

$$C_{a,r,s} = \neg y_a \vee \bigvee_{\substack{b \in R \setminus \{a\} \\ r[b] \neq s[b]}} x_b.$$

Intuitively, the clause $C_{a,r,s}$ represents the fact that if $X \to a$ is a valid, non-trivial FD, then X has to contain at least one attribute b, different from a, such that $r[b] \neq s[b]$. From these clauses, we assemble the subformula

$$\varphi_a = \bigwedge_{\substack{r,s \in \mathfrak{r} \\ r[a] \neq s[a]}} C_{a,r,s}.$$

The output of our reduction is the formula $\varphi = \varphi_{RHS} \wedge \bigwedge_{a \in R} \varphi_a$. Indeed, φ is in conjunctive normal form and has $O(|R|\,|\mathfrak{r}|^2)$ clauses with at most $|R|$ literals each. An encoding of the formula is computable in time linear in its size.

Regarding the correctness of the reduction, recall that we claimed φ to have a weight $k+1$ satisfying assignment iff a non-trivial functional dependency $X \to a$

of size k holds in $\mathtt{r}$. Suppose that the latter is true. We show that setting the variable y_a as well as all x_b with $b \in X$ to TRUE (and all others to FALSE) satisfies φ. Note that the assignment automatically satisfies C_{RHS} and all $C_{b,r,s}$ with $b \neq a$. We are left with the subformula φ_a containing the clauses $C_{a,r,s}$ for row pairs with $r[a] \neq s[a]$. To distinguish these pairs, the LHS X includes, for each of them, some attribute $b \in R \backslash \{a\}$ such that $r[b] \neq s[b]$. Clause $C_{a,r,s}$ is then satisfied by the corresponding literal x_b.

For the other direction, we identify assignments with the variables they set to TRUE. Let $A \subseteq \mathtt{Var}_{LHS} \cup \mathtt{Var}_{RHS}$ be an assignment of Hamming weight $|A| = k+1$ that satisfies φ. The assignment induces two subsets of the schema R, namely, $X = \{a \in R \mid x_a \in A \cap \mathtt{Var}_{LHS}\}$ and $Y = \{a \in R \mid y_a \in A \cap \mathtt{Var}_{RHS}\}$. Due to the clause C_{RHS}, Y is non-empty and X contains at most k elements. Moreover, X and Y are disjoint as the C_a are all satisfied. We say that the generalized functional dependency $X \longrightarrow Y$ holds in a database if $X \longrightarrow a$ holds for every $a \in Y$. It is clearly enough to show that $X \longrightarrow Y$ indeed holds in $\mathtt{r}$.

Assume $X \longrightarrow a$ fails for some $a \in Y$. This is witnessed by a pair of rows $r, s \in \mathtt{r}$ with $r[X] = s[X]$ but $r[a] \neq s[a]$, whence the clause $C_{a,r,s}$ is present in φ. Since $y_a \in A$ is in the assignment, the literal $\neg y_a$ evaluates to FALSE. Also, as X is disjoint from the difference set $\{b \in R \backslash \{a\} \mid r[b] \neq s[b]\}$, no other literal satisfies $C_{a,r,s}$, which is a contradiction. ■

We have established a chain of parameterized reductions between the dependency detection problems of unique column combinations and functional dependencies. Since the endpoints HITTING SET and WEIGHTED 2-NORMALIZED SATISFIABILITY are both $\mathbf{W}[2]$-complete, all of them are, proving Theorem 3.1.

3.2.3 Approximation and Discovery

Our reductions have implications beyond the scope of parameterized complexity. Observe that the transformations in Lemmas 3.5 and 3.6 are computable in polynomial time, do not change the size of the vertex set/relational schema (by more than an additive constant), and preserve approximations with respect to the solution size k or $|X|$, respectively. Also, recall that the minimization version of HITTING SET is **NP**-hard to approximate within a factor of $(1 - \varepsilon) \ln |V|$ for every $\varepsilon > 0$ [DS14]. As a consequence, the minimization versions of UNIQUE COLUMN COMBINATION, FUNCTIONAL DEPENDENCY$_{\textit{fixed RHS}}$, and FUNCTIONAL DEPENDENCY all inherit the same hardness of approximation with respect to the size $|R|$ of the

schema. Previously, an approximation-preserving reduction was only known for the Unique Column Combination problem, starting from Set Cover in [AB96], and for Functional Dependency$_{\textit{fixed RHS}}$ from Dominating Set [CM03].

Rather than approximating minimum solutions, we are mainly interested in the discovery of minimal dependencies in databases. Traditionally, enumeration has been studied via embedded decision problems that are different from those defined in Section 3.2.1. Instead, the Transversal Hypergraph problem (enumerating minimal hitting sets) has been associated with the problem of deciding for two hypergraphs $\mathcal{H}$ and $\mathcal{G}$ whether $\mathcal{G} = \mathrm{Tr}(\mathcal{H})$ or, equivalently, $\mathcal{H} = \mathrm{Tr}(\mathcal{G})$, called the Dual problem. Discover Minimal UCCs analogously corresponds to decide for a database $\mathfrak{r}$ and hypergraph $\mathcal{H}$, whether $\mathcal{H}$ consists of all minimal uniques of $\mathfrak{r}$. Intuitively, this formalizes the decision whether an enumeration algorithm has already found all solutions.

Both decision problems are in **coNP** and it was proven by Eiter and Gottlob [EG95] that they are many-one equivalent.[10] Using a lifting result by Bioch and Ibaraki [BI95], this shows that minimal hitting sets can be enumerated in output-polynomial time if and only if minimal UCCs can, which is the case if and only if Dual is in **P**. Such an equivalence is theoretically appealing and has led to the quasi-output-polynomial upper bound on the running time of hitting set/UCC enumeration that is currently the best known [FK96]. The connection to enumeration has also inspired the intriguing result that Dual is likely not **coNP**-hard as it can be solved in polynomial time when given access to $O(\log^2 N / \log\log N)$ suitably guessed nondeterministic bits [BF99; EGM03; KS03], where N denotes the total input size of the pair $(\mathcal{H}, \mathcal{G})$. There are classes of hypergraphs for which Dual is indeed in **P**, see for example [BGH98; DMP99; EGM03; PS94], or at least in **FPT** with respect to certain structural parameters [EHR08]. The two most well-known special cases that are polynomial-time solvable are the classes of hypergraphs with constant maximum degree [DMP99] or edge size [EGM03], respectively. Translated to our database setting, this means that attributes can only participate in a bounded number of (minimal) difference sets or that any two rows can differ only in a bounded number of columns. Both are severe restrictions and fairly uncommon in real-world databases. For a much more thorough overview of decision problems associated with enumeration, see the recent work by Creignou et al. [Cre+19].

10 Eiter and Gottlob [EG95] in fact consider the complementary problem, whether $\mathfrak{r}$ admits a unique column combination *not* in $\mathcal{H}$, under the name Additional Key.

Unfortunately, the approach described above holds only limited value when it comes to designing practical algorithms. Imagine an implementation of the discovery of minimal UCCs of a database $\mathfrak{r}$ via repeated checks whether the hypergraph $\mathcal{H}$ of previously found solutions is already complete. Such an algorithm is bound to use an amount of memory that is exponential in the size of $\mathfrak{r}$. This is due to the fact that some databases have exponentially many minimal solutions and the decision subroutine at least has to read all of $\mathcal{H}$. Note that such a large memory consumption is not at all necessary as there are algorithms known for Transversal Hypergraph whose space complexity is only linear in the input size [EHR08; MU14]. In fact, enumeration algorithms are often analyzed not only with respect to their running time, but also in terms of space consumption, see [Con+20; CS19]. For data profiling problems like Discover Minimal UCCs on the other hand, space-efficient algorithms have only recently started to received some attention. We give two examples in Chapters 4 and 7 of this thesis, another one is due to Köhler et al. [Köh+16].

In the following, we simplify and thereby extend the above equivalences making them usable in practice, namely, we prove Theorem 3.3. It states the existence of parsimonious reductions, in both directions, that relate the input instances directly on the level of the enumeration problems, without decision problems as intermediaries. This way, we characterize the enumeration complexity of unique column combinations as well as functional dependencies, both with fixed and arbitrary right-hand side. It is worth noting that our insights on enumeration do stem from the study of decision problems, but the results are entirely lifted.

The reductions between the decision problems Hitting Set, Unique Column Combination, Functional Dependency$_{\textit{fixed RHS}}$, and Functional Dependency (in that order) for *minimum* dependencies described in Lemmas 3.5 and 3.6 are all built on bijective correspondences between the solutions. The running times of the reductions are polynomial and independent of the given budget k. Finally, the mappings of the solution spaces also preserve set inclusions. This means, the same input transformations applied to the discovery of *minimal* dependencies are in fact parsimonious reductions from Transversal Hypergraph to Discover Minimal UCCs and onward to Discover Minimal FDs/FDs$_{\textit{fixed RHS}}$. For the inverse direction, Observation 2.2 shows that the enumeration of UCCs is at most as hard as that of hitting sets, that is, Discover Minimal UCCs and Transversal Hypergraph are equivalent. It is worth noting that the parsimonious reductions increase the size of the instances at most by a polynomial factor (in most cases a

$V = \{a, b, c, d, e\}$

$\mathcal{H}_1 = \{\{a, b\}, \{b, c\}\}$

$\mathcal{H}_2 = \{\{b, c\}, \{b, d, e\}\}$

$\mathcal{H}_3 = \{\{a, c\}, \{a, d, e\}, \{b\}\}$

	a	b	c	d	e	x_1	x_2	x_3
r_0	0	0	0	0	0	0	0	0
	1	1	0	0	0	1	0	0
r_2	0	2	2	0	0	1	0	0
r_3	0	3	3	0	0	0	1	0
	0	4	0	4	4	0	1	0
	5	0	5	0	0	0	0	1
	6	0	0	6	6	0	0	1
	0	7	0	0	0	0	0	1
r_a	×	0	0	0	0	0	0	0
	0	×	0	0	0	0	0	0
	0	0	×	0	0	0	0	0
	0	0	0	×	0	0	0	0
	0	0	0	0	×	0	0	0

Figure 3.4: Illustration of Lemma 3.8. The three hypergraphs $\mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3$ on the vertex set V are on the left and the equivalent database $\mathfrak{r}$ is on the right. The hypergraphs $\mathcal{H}_1$ and $\mathcal{H}_2$ share the edge $\{b, c\}$, but this results in two rows r_2 and r_3. The corresponding transversal hypergraphs are $\mathrm{Tr}(\mathcal{H}_1) = \{\{a, c\}, \{b\}\}$, $\mathrm{Tr}(\mathcal{H}_2) = \{\{b\}, \{c, b\}, \{c, e\}\}$, and $\mathrm{Tr}(\mathcal{H}_3) = \{\{a, b\}, \{b, c, d\}, \{b, c, e\}\}$. The functional dependencies $b \to x_1$ and $b \to x_2$ indeed hold in $\mathfrak{r}$, and adding the attribute a further gives $ab \to x_3$. The rows in the last block eliminate all trivial functional dependencies $X \to v$ for $v \in V$.

constant one) *in the input size only* and therefore transfer the space complexity of any enumeration algorithm from one side to the other.

To complete the proof of Theorem 3.3, we still need the following lemma that characterizes the complexity of functional dependency discovery in terms of the Transversal Hypergraph Union problem.

▶ **Lemma 3.8.** Discover Minimal FDs and Transversal Hypergraph Union are equivalent under parsimonious reductions. Moreover, there is a parsimonious reduction from Discover Minimal FDs$_{\textit{fixed RHS}}$ to Transversal Hypergraph. ◀

Proof. This proof uses techniques that already helped to establish the previous lemmas of this section. Let $\mathfrak{r}$ be a database over schema R, $a \in R$ some attribute, $r, s \in \mathfrak{r}$ two rows with $r[a] \neq s[a]$. Recall that their difference set is $D(r, s) = \{b \in R \mid r[b] \neq s[b]\}$. We define their *punctured difference set* to be $D(r, s) \backslash \{a\}$. It is implicit in the proof of Lemma 3.7–and easy to verify from the definition–that a set $X \subseteq R \backslash \{a\}$ is the left-hand side of a valid, minimal, non-trivial FD

$X \to a$ if and only if it is a minimal hitting set for the hypergraph of punctured difference sets $\mathcal{D}_a = \{D(r,s)\backslash\{a\} \mid r,s \in \mathfrak{r}; r[a] \neq s[a]\}$.

Transforming the input database $\mathfrak{r}$ over schema $R = \{a_1, \ldots, a_{|R|}\}$ into the $|R|$ hypergraphs $\mathcal{D}_{a_1}, \ldots, \mathcal{D}_{a_{|R|}}$ gives a parsimonious reduction from the Discover Minimal FDs problem to Transversal Hypergraph Union. In the same fashion, fixing the desired right-hand side in the input reduces Discover Minimal FDs$_{\textit{fixed RHS}}$ to the Transversal Hypergraph problem.

The opposite reduction is the main part of the lemma. We are given hypergraphs $\mathcal{H}_1, \ldots, \mathcal{H}_d$, without loss of generality all on the same vertex set V, and we need to compute some database $\mathfrak{r}$ such that its valid, non-trivial functional dependencies are in one-to-one correspondence with the hitting sets of the $\mathcal{H}_i$. An example can be seen in Figure 3.4. As the relational schema, we take the set $R = V \cup \{x_1, \ldots, x_d\}$, where the x_i are attributes not previously appearing in V. The construction of $\mathfrak{r}$ starts similarly to Lemma 3.5. We first add the all-zeros row r_0 (with $r_0[a] = 0$ for every $a \in R$). Let $m = \sum_{i=1}^{d} |\mathcal{H}_i|$ be the total number of edges and $E_1, E_2, \ldots, E_m$ an arbitrary numbering of them. Note that if the same set of vertices is an edge of multiple hypergraphs, it appears in the list with that multiplicity. For every edge E_j, we add the following row r_j,

$$r_j[a] = \begin{cases} j, & \text{if } a \in E_j; \\ 0, & \text{if } a \in V\backslash E_j; \\ 1, & \text{if } a = x_i \text{ such that } E_j \in \mathcal{H}_i; \\ 0, & \text{otherwise.} \end{cases}$$

In other words, the subtuple $r_j[V]$ is the characteristic vector of E_j only that its non-zero entries are j instead of 1; the subtuple $r_j[\{x_1, \ldots, x_d\}]$ has a one 1 at the position corresponding to the hypergraph containing E_j and 0 everywhere else. The remaining construction of database $\mathfrak{r}$ uses an idea of Lemma 3.6 (*ii*). Let $\times$ be a new symbol. For every vertex $v \in V$, we add the row r_v with $r_v[v] = \times$ and $r_v[a] = 0$ for all other attributes $a \in R\backslash\{v\}$. The database $\mathfrak{r}$ can be obtained in time polynomial in m and $|V|$.

We claim that the minimal, valid, non-trivial functional dependencies of $\mathfrak{r}$ are exactly those having the form $T \to x_i$ with $T \in \mathrm{Tr}(\mathcal{H}_i)$. The existence of a parsimonious reduction from Transversal Hypergraph Union to Discover Minimal FDs easily follows from that. Let $X \subseteq R$ and $a \in R\backslash X$ be such that the FD $X \to a$ holds in $\mathfrak{r}$ and is minimal. For any $v \in V$, the rows r_0 and r_v differ

only in attribute v, therefore v is not the right-hand side of any non-trivial FD, whence $a = x_i$ for some $1 \leq i \leq d$. As seen above, the set X must be a minimal transversal of the hypergraph $\mathcal{D}_{x_i} = \{D(r,s)\backslash\{x_i\} \mid r, s \in \mathfrak{r}; r[x_i] \neq s[x_i]\}$. We are left to prove that $\mathcal{D}_{x_i}$ has the same minimal transversals as $\mathcal{H}_i$. Let $r, s \in \mathfrak{r}$ be rows that differ in attribute x_i, say, $r[x_i] = 1$ and $s[x_i] = 0$. We thus have $r = r_j$ for some $1 \leq j \leq m$. The rows r and s share only the value 0, if any. Therefore,

$$D(r,s) = \begin{cases} E_j \cup \{x_i\}, & \text{if } s = r_0; \\ E_j \cup E_k \cup \{x_i, x_\ell\}, & \text{if } s = r_k \text{ for } 1 \leq k \leq m \text{ such that } E_k \in \mathcal{H}_\ell; \\ E_j \cup \{x_i, v\}, & \text{if } s = r_v \text{ for } v \in V. \end{cases}$$

In the second case, note that $\ell \neq i$ since $s[x_i] = 0$. The above implies that $\mathcal{H}_i \subseteq \mathcal{D}_{x_i}$; moreover, all edges in $\mathcal{D}_{x_i}\backslash\mathcal{H}_i$ are supersets of ones in $\mathcal{H}_i$. The respective minimizations $\min(\mathcal{D}_{x_i}) = \min(\mathcal{H}_i)$ are thus equal and, by duality, also their transversal hypergraphs $\mathrm{Tr}(\mathcal{D}_{x_i}) = \mathrm{Tr}(\mathcal{H}_i)$ are the same. In particular, this also shows that every FD of the form $T \longrightarrow x_i$ for $T \in \mathrm{Tr}(\mathcal{D}_{x_i})$ is valid and minimal (and of course non-trivial) in $\mathfrak{r}$. ■

Discover Minimal FDs can also be solved in output-polynomial time (incrementally polynomially/with polynomial delay) if and only if Transversal Hypergraph can [EG95], this has been established along the same lines as discussed in the remarks preceding Lemma 3.8. We leave it as an open problem to give a parsimonious reduction. We have shown above that this is equivalent to encoding the hitting set information of $|R|$ different hypergraphs in a single one.

Notably, Eiter and Gottlob [EG95] additionally presented an alternative construction that is *almost* parsimonious. The only condition they needed to relax is the bijection between the solution spaces. They transformed a database over schema R into some hypergraph $(R^2, \mathcal{F})$ such that most of its minimal hitting sets indeed correspond to the functional dependencies with arbitrary RHS. Additionally, $\mathcal{F}$ has some $O(|R|^4)$ excess solutions, that is, polynomially many in the input size only, which do not have an FD counterpart, but are easily recognizable. We include their result here since the extended version of [EG95] does not seem to be widely available and we feel that the techniques used in the proof might be helpful to establish a fully parsimonious reduction. We discuss the construction entirely in the language of hitting sets as justified by Lemma 3.8.

▶ **Proposition 3.9 (Theorem 7.6 in the extended version of [EG95]).** Let $\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_d$ be hypergraphs, all on the same vertex set V. There exist hypergraphs $\mathcal{F}$ and $\mathcal{G}$ on the vertex set $V \times [d]$ and a bijective function

$$g\colon \operatorname{Tr}(\mathcal{F}) \to \operatorname{Tr}(\mathcal{H}_1) \dot\cup \operatorname{Tr}(\mathcal{H}_2) \dot\cup \ldots \dot\cup \operatorname{Tr}(\mathcal{H}_d) \dot\cup \mathcal{G}$$

such that the following three statements hold.

(*i*) $\mathcal{F}$ is computable from $V, \mathcal{H}_1, \ldots, \mathcal{H}_d$ in polynomial time;

(*ii*) $\mathcal{G}$ is of size $|\mathcal{G}| \leq \binom{d}{2} \cdot |V|^2$;

(*iii*) $g(S)$ is computable in time linear in $|S|$ for any $S \in \operatorname{Tr}(\mathcal{F})$. ◀

Proof. We fix some notation for the pairs in $V \times [d]$. We use their second component to indicate the hypergraph of origin. For some index $i \in [d]$ and edge $E \in \mathcal{H}_i$, we let $E' = \{(v, i) \mid v \in E\}$ be the result of augmenting the index as the second component; the same with $T \in \operatorname{Tr}(\mathcal{H}_i)$, for which we define $T' = \{(v, i) \mid v \in T\}$. Conversely, for some set $S \subseteq V \times [d]$, let $\pi_1(S) = \{v \in V \mid \exists i \in [d]\colon (v, i) \in S\}$ be the projection onto the first component and $\pi_2(S) = \{i \in [d] \mid \exists v \in V\colon (v, i) \in S\}$ the projection onto the second component.

We first construct the hypergraph $\mathcal{F}$. Fix some $i \in [d]$ and $E \in \mathcal{H}_i$. Given the augmented edge E', add all possible pairs with second component different from i to obtain $E'' = E' \cup \{(v, j) \mid v \in V; j \in [d] \backslash \{i\}\}$. The intermediate hypergraphs $\mathcal{F}_i = \{E'' \mid E \in \mathcal{H}_i\}$ consist of all edges defined this way and $\mathcal{F} = \bigcup_{i \in [d]} \mathcal{F}_i$ is their union. Clearly, $\mathcal{F}$ can be computed in time polynomial in the input.

Assume for the moment that all hitting sets of $\mathcal{H}_i$, for every i, have at least two elements. Define $\mathcal{G} = \{\{(v, i), (u, j)\} \mid u, v \in V; 1 \leq i < j \leq d\}$, it has $\binom{d}{2}|V|^2$ edges. We claim that the minimal hitting sets of $\mathcal{F}$ can be partitioned as

$$\operatorname{Tr}(\mathcal{F}) = \{T' \mid T \in \operatorname{Tr}(\mathcal{H}_1)\} \dot\cup \ldots \dot\cup \{T' \mid T \in \operatorname{Tr}(\mathcal{H}_d)\} \dot\cup \mathcal{G}.$$

In particular, the solutions in $\operatorname{Tr}(\mathcal{F})$ carry the information to which input hypergraph they belong. Note that this implies that the desired linear-time computable bijection is

$$g(S) = \begin{cases} \pi_1(S), & \text{if } |\pi_2(S)| = 1; \\ S, & \text{otherwise.} \end{cases}$$

In order to prove the claim, let $T \in \operatorname{Tr}(\mathcal{H}_i)$ be a minimal transversal. The augmented set T' hits all of the E' for $E \in \mathcal{H}_i$ and therefore all edges of $\mathcal{F}_i$.

Any pair $(v, i) \in T'$ alone suffices to hit the edges contained in any of the other hypergraphs $\mathcal{F}_j$ with $j \neq i$. To show that T' is in fact minimal, let $v \in T$ be a vertex and $E_v \in \mathcal{H}_i$ its private edge as guaranteed by Observation 2.1. Removing the pair (v, i) from T' would result in edge E''_v being unhit. Therefore, $T' \in \mathrm{Tr}(\mathcal{F})$ is minimal. Also, $\{(v, i), (u, j)\} \in \mathcal{G}$ hits all edges of $\mathcal{F}$: vertex (v, i) was added to all edges in $\mathcal{F}_j$, $j \neq i$, and (u, j) to the ones in $\mathcal{F}_i$. By the assumption that neither $\mathcal{H}_i$ nor $\mathcal{H}_j$ admit singleton transversals, those sets are minimal as well.

Conversely, let $S \subseteq V \times [d]$ be any hitting set for $\mathcal{F}$. We show that S contains an edge of $\mathcal{G}$ or of some $\{T' \mid T \in \mathrm{Tr}(\mathcal{H}_i)\}$. If $|\pi_2(S)| \geq 2$, the first case is true. Otherwise, we have $\pi_2(S) = \{i\}$ for some $i \in [d]$ and S intersects all edges of $\mathcal{F}_i$ in their E'-part. Thus, S contains an augmented transversal T' of $T \in \mathrm{Tr}(\mathcal{H}_i)$.

Finally, if there is an index $i \in [d]$ and a vertex $v \in V$ such that the singleton $\{v\} \in \mathrm{Tr}(\mathcal{H}_i)$ alone is a minimal hitting set for $\mathcal{H}_i$, then we remove all edges $\{(v, i), (u, j)\}$ for $u \in V$ and $j \in [d] \setminus \{i\}$ from $\mathcal{G}$ to preserve minimality. ■

3.3 Inclusion Dependencies

We now discuss inclusion dependencies in relational databases. We show that their detection problem, when parameterized by the solution size, is one of the first natural problems to be complete for the class $\mathbf{W}[3]$. We do so by proving its FPT-equivalence with the weighted satisfiability problem for a certain fragment of propositional logic. Later in Section 3.3.4, we show how to transfer our results to the discovery of maximal inclusion dependencies.

3.3.1 Problem Definition

A Boolean formula whose NOT-gates (if any) appear only immediately after an input node is *antimonotone* if *all* inputs are negated. We identify a variable assignment with the set of those variables that are assigned TRUE. In the case of antimonotone formulas, this means that the satisfying assignments are closed under arbitrarily turning variables to FALSE, that is, taking subsets. The empty assignment that assigns FALSE to all variables is always satisfying. Recall from Section 2.4.2 that a formula is 3-normalized if it is a conjunction of disjunctions of conjunctions of literals or, equivalently, if it is a conjunction of subformulas in disjunctive normal form (DNF). An example formula is

$$((\neg x_1 \wedge \neg x_2 \wedge \neg x_4) \vee (\neg x_3 \wedge \neg x_4)) \wedge ((\neg x_1 \wedge \neg x_3) \vee (\neg x_2 \wedge \neg x_5) \vee (\neg x_1 \wedge \neg x_4 \wedge \neg x_5)).$$

It admits satisfying assignments of Hamming weight 0, 1, and 2, but none of larger weight. We used the Weighted 3-normalized Satisfiability problem to define the class **W**[3]. Weighted Antimonotone 3-normalized Satisfiability is the special case in which the inputs are restricted to antimonotone formulas.

Weighted Antimonotone 3-normalized Satisfiability (WA3NS)

Instance: An antimonotone, 3-normalized Boolean formula φ and a non-negative integer k.

Parameter: The non-negative integer k.

Decision: Does φ admit a satisfying assignment of Hamming weight k?

By the above remark, this is the same as asking for an assignment of weight at least k. The Antimonotone Collapse Theorem of Downey and Fellows [DF95a; DF95b] implies that the WA3NS special case is **W**[3]-complete in its own right.

The inclusion-wise maximal satisfying assignments carry the full information about the collection of all satisfying assignments. It is therefore natural to define the corresponding enumeration problem as follows.

Enumerate Maximal Satisfying WA3NS Assignments

Instance: An antimonotone, 3-normalized Boolean formula φ.

Enumeration: List all maximal satisfying assignments of φ.

For inclusion dependencies, the situation is similar. Every subset of a valid IND is also valid. Asking for a dependency of size exactly k is thus the same as asking for one of size at least k. We define two variants of the decision problem, similar as we did with functional dependencies. The more restricted variant requires the two databases to have the same schema with the identity mapping between columns.

Inclusion Dependency$_{Identity}$

Instance: Two relational databases $\mathfrak{r}$, $\mathfrak{s}$ over schema R and a non-negative integer k.

Parameter: The non-negative integer k.

Decision: Is there a set $X \subseteq R$ with $|X| = k$ such that $\mathfrak{r}[X] \subseteq \mathfrak{s}[X]$ is an inclusion dependency?

Inclusion Dependency

Instance: Two relational databases, $\mathfrak{r}$ over schema R and $\mathfrak{s}$ over S, and a non-negative integer k.

Parameter: The non-negative integer k.

Decision: Is there a set $X \subseteq R$ with $|X| = k$ and an injective mapping $\sigma\colon X \to S$ such that $\mathfrak{r}[X] \subseteq \mathfrak{s}[\sigma(X)]$ is an inclusion dependency?

The unparameterized variant of the general Inclusion Dependency problem is **NP**-complete[11] already for pairs of binary databases [Kan+92].

The solutions of Inclusion Dependency$_{Identity}$ are mere subsets of the underlying schema, therefore it is clear what we mean by a maximal solution. The case of the general Inclusion Dependency problem is slightly more intricate. Recall from Section 2.3.3 that we say a general inclusion dependency (X, σ) is maximal if there is no other IND (X', σ') such that $X \subsetneq X'$ is a proper subset and σ is the restriction of σ' to X. Note that the pair (X', τ) with an alternative mapping τ might still be a valid inclusion dependency. This leads to the following enumeration problems.

Discover Maximal INDs$_{Identity}$

Instance: Two relational databases $\mathfrak{r}$, $\mathfrak{s}$ over the same schema.

Enumeration: List all maximal valid inclusion dependencies between $\mathfrak{r}$ and $\mathfrak{s}$ with the identity mapping between the columns.

Discover Maximal INDs

Instance: Two relational databases $\mathfrak{r}$ and $\mathfrak{s}$.

Enumeration: List all maximal valid inclusion dependencies between $\mathfrak{r}$ and $\mathfrak{s}$.

3.3.2 Membership in $W[3]$

We show that both variants of the Inclusion Dependency decision problem are contained in the class **W**[3]. Recall that Inclusion Dependency$_{Identity}$ restricts the input to pairs $(\mathfrak{r}, \mathfrak{s})$ of databases over the same schema and forbids solutions

11 Kantola et al. [Kan+92] define Full IND Existence as the special case of Inclusion Dependency with $k = |R|$ and show its **NP**-completeness.

in which the set of values $\mathfrak{r}[a]$ of one column are contained in $\mathfrak{s}[b]$ for some other column $b \neq a$. As a first step, we show that (not entirely surprisingly) this variant is at most as hard as the general problem.

▶ **Lemma 3.10.** There is a parameterized reduction from the Inclusion Dependency$_{Identity}$ problem to Inclusion Dependency. ◀

Proof. Let $\mathfrak{r}$ and $\mathfrak{s}$ be two databases over the schema R and let $t^{\times} = (\times_a)_{a \in R}$ be a new row, where the $\times_a$ are $|R|$ different symbols none of which are previously used anywhere in $\mathfrak{r}$ or $\mathfrak{s}$. In the restricted setting, an inclusion dependency is a set $X \subseteq R$ of columns such that $\mathfrak{r}[X] \subseteq \mathfrak{s}[X]$. It is easy to see that $(\mathfrak{r}, \mathfrak{s})$ has such an inclusion dependency of size k, for any k, if and only if $(\mathfrak{r} \cup \{t^{\times}\}, \mathfrak{s} \cup \{t^{\times}\})$ has an inclusion dependency of the same size with an arbitrary mapping between the columns since $\times_a \in \mathfrak{s}[b]$ holds iff $a = b$. The lemma follows from here. ■

To demonstrate the membership of the general problem in $\mathbf{W}[3]$, we reduce is to WA3NS. Namely, we compute from the two databases an antimonotone, 3-normalized formula which has a weight k satisfying assignment if and only if the databases admit an inclusion dependency of that cardinality. For this, we use a correspondence between *pairs* of attributes and Boolean variables.

▶ **Lemma 3.11.** There is a parameterized reduction from Inclusion Dependency to Weighted Antimonotone 3-normalized Satisfiability. ◀

Proof. Let $R = \{a_1, \ldots, a_{|R|}\}$ and $S = \{b_1, \ldots, b_{|S|}\}$ be two schemas. We introduce a Boolean variable $x_{i,j}$ for each pair of attributes $a_i \in R$ and $b_j \in S$. We let $\mathtt{Var}_P$ denote the set of variables corresponding to a collection $P \subseteq R \times S$ of such pairs. Consider a subset $X \subseteq R$ together with an injection $\sigma \colon X \to S$. From this, we construct a truth assignment including the variable $x_{i,j}$ (setting it to true) iff $a_i \in X$ and $\sigma(a_i) = b_j$. The resulting assignment has weight $|X|$ and the collection of all possible configurations (X, σ) is uniquely described by $\mathtt{Var}_{R \times S}$ and the truth assignments obtained this way. Moreover, these assignments all satisfy the following antimonotone Boolean formula φ_{map}.

$$\varphi_{map} = \left(\bigwedge_{i=1}^{|R|} \bigwedge_{j=1}^{|S|-1} \bigwedge_{j'=j+1}^{|S|} (\neg x_{i,j} \vee \neg x_{i,j'}) \right) \quad \wedge \quad \left(\bigwedge_{j=1}^{|S|} \bigwedge_{i=1}^{|R|-1} \bigwedge_{i'=i+1}^{|R|} (\neg x_{i,j} \vee \neg x_{i',j}) \right).$$

The first half of φ_{map} expresses that, for every pair of variables $x_{i,j}$ and $x_{i,j'}$ with $j \neq j'$, at most one of them shall be true; the second half is satisfied if the same

holds for all pairs $x_{i,j}$ and $x_{i',j}$ with $i \neq i'$. Conversely, a satisfying assignment A (a subset of $\mathtt{Var}_{R\times S}$) for φ_{map} defines a relation $\sigma \subseteq R \times S$ and a set $X \subseteq R$ by setting $\sigma = \{(a_i, b_j) \mid x_{i,j} \in A\}$ and $X = \{a_i \in R \mid \exists 1 \leq j \leq |S| \colon x_{i,j} \in A\}$. By construction, the relation σ is not only a function $\sigma \colon X \to S$, but an injection. In summary, φ_{map} is fulfilled exactly by the assignments described above. Observe that φ_{map} is in CNF and therefore also 3-normalized as each literal is a conjunctive clause of length 1.

We now formalize the requirement that a configuration (X, σ) is an inclusion dependency in a given pair of databases $\mathfrak{r}$ and $\mathfrak{s}$ over the respective schemas R and S, that is, that $\mathfrak{r}[X] \subseteq \mathfrak{s}[\sigma(X)]$ holds. First, assume that each database consists only of a single row r_ℓ and s_m, respectively. We say a pair of attributes $(a_i, b_j) \in R \times S$ is *forbidden* for r_ℓ and s_m if $r_\ell[a_i] \neq s_m[b_j]$. Let $F_{\ell,m}$ be the set of all forbidden pairs. For an configuration (X, σ) to be an IND, the variables $x_{i,j}$ need to be set to FALSE for all $(a_i, b_j) \in F_{\ell,m}$. In terms of Boolean formulas, this is represented by the conjunctive clause[12] $M_{\ell,m} = \bigwedge_{x \in \mathtt{Var}_{F_{\ell,m}}} \neg x$. It follows that (X, σ) is an inclusion dependency if and only if the corresponding variable assignment satisfies both φ_{map} and $M_{\ell,m}$.

Now suppose $\mathfrak{s}$ has multiple rows, while $\mathfrak{r}$ is still considered to have only one. The configuration (X, σ) is an IND for $(\mathfrak{r}, \mathfrak{s})$ iff it is one for at least one instance $(\mathfrak{r}, \{s_m\})$ with $s_m \in \mathfrak{s}$. If also $\mathfrak{r}$ has more records, then (X, σ) is an IND for $(\mathfrak{r}, \mathfrak{s})$ iff it is one in each instance $(\{r_\ell\}, \mathfrak{s})$ with $r_\ell \in \mathfrak{r}$. Therefore, we obtain an inclusion dependency if and only if φ_{map} and the formula

$$\varphi = \bigwedge_{r_\ell \in \mathfrak{r}} \bigvee_{s_m \in \mathfrak{s}} M_{\ell,m}$$

are simultaneously satisfied by the assignment corresponding to (X, σ).

The formula $\varphi \wedge \varphi_{map}$ is antimonotone and 3-normalized. The (disjunctive) clauses of φ_{map} can be constructed in total time $O(|R|^2|S| + |R||S|^2)$ and all sets $F_{\ell,m}$ together are computable in time $O(|\mathfrak{r}||\mathfrak{s}||R||S|)$. An encoding of $\varphi \wedge \varphi_{map}$ can thus be obtained from the input databases $\mathfrak{r}$ and $\mathfrak{s}$ in polynomial time. Finally, by the above observation that any solution for the sub-formula φ_{map} that corresponds to (X, σ) has weight $|X|$, the reduction preserves the parameter. ■

12 Conjunctive clauses are sometimes also called *monoms* [Weg91].

3.3.3 Hardness for $W[3]$

We now show that detecting inclusion dependencies is also hard for **W**[3]. We argue that the existence of weighted satisfying assignments for 3-normalized, antimonotone formulas can be decided by solving instances of the more restricted Inclusion Dependency$_{Identity}$ variant. For this, we make use of indicator functions. On the one hand, we interpret propositional formulas φ over n variables as Boolean functions $f_\varphi\colon \{0,1\}^n \to \{0,1\}$ in the obvious way. On the other hand, for a pair of databases $\mathfrak{r}$ and $\mathfrak{s}$ over the same schema R, we represent any subset $X \subseteq R$ by its characteristic vector of length $|R|$. We then define the *indicator function* $f_{(\mathfrak{r},\mathfrak{s})}\colon \{0,1\}^{|R|} \to \{0,1\}$ by requiring that $f_{(\mathfrak{r},\mathfrak{s})}(X) = 1$ holds iff X is an inclusion dependency (with the identity mapping between the columns).

We claim that for any formula φ that is antimonotone and 3-normalized, there is a pair $(\mathfrak{r},\mathfrak{s})$ of databases computable in polynomial time such that $f_\varphi = f_{(\mathfrak{r},\mathfrak{s})}$. Clearly, this gives a parameterized reduction from WA3NS to the Inclusion Dependency$_{Identity}$ problem. The remainder of this section is dedicated to proving this claim. Recall that the top-level connective of a 3-normalized formula is a conjunction. We start by demonstrating how to model this using databases.

▶ **Lemma 3.12.** Let $(\mathfrak{r}^{(1)},\mathfrak{s}^{(1)})$ and $(\mathfrak{r}^{(2)},\mathfrak{s}^{(2)})$ be two pairs of databases, all over the same schema R, with indicator functions $f^{(1)}$ and $f^{(2)}$, respectively. There exists a polynomial time computable pair $(\mathfrak{r},\mathfrak{s})$ (over R) of size $|\mathfrak{r}| = |\mathfrak{r}^{(1)}| + |\mathfrak{r}^{(2)}|$ and $|\mathfrak{s}| = |\mathfrak{s}^{(1)}| + |\mathfrak{s}^{(2)}|$, having indicator function $f_{(\mathfrak{r},\mathfrak{s})} = f^{(1)} \wedge f^{(2)}$. ◀

Proof. Without loosing generality, the values appearing in $\mathfrak{r}^{(1)}$ and $\mathfrak{s}^{(1)}$ are disjoint from those in $\mathfrak{r}^{(2)}$ and $\mathfrak{s}^{(2)}$. We straightforwardly construct $(\mathfrak{r},\mathfrak{s})$ as $\mathfrak{r} = \mathfrak{r}^{(1)} \cup \mathfrak{r}^{(2)}$ and $\mathfrak{s} = \mathfrak{s}^{(1)} \cup \mathfrak{s}^{(2)}$, which matches the requirements on both the computability and size. We still need to show $f_{(\mathfrak{r},\mathfrak{s})} = f^{(1)} \wedge f^{(2)}$.

Equivalently, we prove that a set $X \subseteq R$ is an inclusion dependency in $(\mathfrak{r},\mathfrak{s})$ if and only if it is one in both pairs $(\mathfrak{r}^{(1)},\mathfrak{s}^{(1)})$ and $(\mathfrak{r}^{(2)},\mathfrak{s}^{(2)})$. Let X be an IND in $(\mathfrak{r}^{(1)},\mathfrak{s}^{(1)})$ as well as $(\mathfrak{r}^{(2)},\mathfrak{s}^{(2)})$. That means, for every row $r \in \mathfrak{r}^{(1)}$, there exists some $s \in \mathfrak{s}^{(1)}$ with $r[X] = s[X]$; same for $\mathfrak{r}^{(2)}$ and $\mathfrak{s}^{(2)}$. As all those rows are also present in $(\mathfrak{r},\mathfrak{s})$, X is an IND there as well. Conversely, suppose X is not an inclusion dependency in, say, $(\mathfrak{r}^{(1)},\mathfrak{s}^{(1)})$. Then, $\mathfrak{r}^{(1)}$ has a row r that disagrees with every $s \in \mathfrak{s}^{(1)}$ on some attribute in X. The record r is also in present in $\mathfrak{r}$ and all rows in $\mathfrak{s}$ belong either to $\mathfrak{s}^{(1)}$ or have completely disjoint values. This results in $r[X] \neq s[X]$ for *every* record $s \in \mathfrak{s}$, as desired. ■

$\varphi = M_1 \vee M_2 \vee M_3$

$M_1 = (\neg x_1 \wedge \neg x_2 \wedge \neg x_3)$
$M_2 = (\neg x_2 \wedge \neg x_4 \wedge \neg x_5)$
$M_3 = (\neg x_1 \wedge \neg x_3 \wedge \neg x_4 \wedge \neg x_6)$

a_1	a_2	a_3	a_4	a_5	a_6
1	1	1	0	0	0
0	2	0	2	2	0
3	0	3	3	0	3

a_1	a_2	a_3	a_4	a_5	a_6
×	×	×	0	0	0
×	×	×	2	2	0
×	×	×	3	0	3
1	×	1	×	×	0
0	×	0	×	×	0
3	×	3	×	×	3
×	1	×	×	0	×
×	2	×	×	2	×
×	0	×	×	0	×

Figure 3.5: Illustration of Lemma 3.13. The antimonotone DNF formula φ on the left has the three conjunctive clauses M_1, M_2, M_3. The equivalent instance of Inclusion Dependency$_{Identity}$ consists of database $\mathfrak{r}$ in the center and $\mathfrak{s}$ on the right. There are three maximal inclusion dependencies $\{a_4, a_5, a_6\}$, $\{a_1, a_3, a_6\}$, and $\{a_2, a_5\}$. Adding any more attributes to either of them would create a hitting set for the conjunctive clauses, corresponding to an unsatisfying assignment.

One could hope that there is a method that translates disjunctions to the database domain in a similar fashion as above. However, we believe that there is none that is both computable in FPT-time and compatible with a complementing way of representing conjunctions. The reason is as follows. Negative literals are easily expressed by pairs of single-row databases. Together with FPT-time procedures of constructing conjunctions as well as disjunctions, one could encode antimonotone Boolean formulas of arbitrary logical depth. The Antimonotone Collaps Theorem [DF13], states that the Weighted Antimonotone t-normalized Satisfiability problem is $\mathbf{W}[t]$-complete for every odd $t \geq 3$. This would then render Inclusion Dependency$_{Identity}$ to be hard for all classes $\mathbf{W}[t]$ and, as a consequence of Lemmas 3.10 and 3.11, the $\mathbf{W}$-hierarchy would collapse to its third level. That being said, there is a method specifically tailored to antimonotone DNF formulas.

▶ **Lemma 3.13.** Let φ be an antimonotone formula in disjunctive normal form. There are relational databases $\mathfrak{r}$ and $\mathfrak{s}$ over the same schema computable in time polynomial in the size of φ such that $f_\varphi = f_{(\mathfrak{r},\mathfrak{s})}$. ◀

Proof. Let $x_1, \ldots, x_n$ be the variables of φ. Define the schema $R = \{a_1, \ldots, a_n\}$ by identifying variable x_i with attribute a_i. We first describe the database $\mathfrak{r}$

and subsequently construct a matching database $\mathfrak{s}$. Each of the m constituting conjunctive clauses $M_1, \ldots, M_m$ of the DNF formula φ is represented by some row in $\mathfrak{r}$, see Figure 3.5. Similar to the proof of Lemma 3.5, define row r_j as

$$r_j[a_i] = \begin{cases} j, & \text{if } x_i \text{ occurs in clause } M_j; \\ 0, & \text{otherwise.} \end{cases}$$

The second database $\mathfrak{s}$ is constructed by first creating m copies of $\mathfrak{r}$. Let $\times$ be a new symbol not appearing anywhere in $\mathfrak{r}$. In the j-th copy of $\mathfrak{r}$, we set the value for attribute a_i to $\times$ whenever variable x_i occurs in M_j. (See Figure 3.5 again.) Note that $|R| = n$ equals the number of variables of φ and $|\mathfrak{r}|$ is linear in the number m of conjunctive clauses, while $|\mathfrak{s}|$ is quadratic. The time to compute the pair $(\mathfrak{r}, \mathfrak{s})$ is linear in their combined size and polynomial in the size of φ. It is left to show that the indicator function satisfies $f_\varphi = f_{(\mathfrak{r},\mathfrak{s})}$.

First, suppose $f_\varphi(X) = 1$ for some length-n binary vector X or, equivalently, for some subset $X \subseteq R$. We show that $f_{(\mathfrak{r},\mathfrak{s})}(X) = 1$, meaning that X is an inclusion dependency in $(\mathfrak{r}, \mathfrak{s})$. Necessarily, we have $f_{M_j}(X) = 1$ for at least one conjunctive clause M_j. Since M_j contains only negative literals, all of its variables evaluate to FALSE. This is equivalent to X not containing any attribute that corresponds to a variable in M_j. In the j-th copy of $\mathfrak{r}$ in the database $\mathfrak{s}$, the values were changed to $\times$ for exactly those attributes. Thus, the projection $\mathfrak{s}[X]$ contains an exact copy of $\mathfrak{r}[X]$ and X is indeed an IND, resulting in $f_{(\mathfrak{r},\mathfrak{s})}(X) = 1$.

For the opposite direction, suppose $f_\varphi(X) = 0$. Each conjunctive clause thus contains a variable corresponding to some attribute in X. Consequently, in each row of $\mathfrak{s}$, there is an attribute in X whose value was replaced by $\times$. As $\mathfrak{r}$ does not contain the symbol $\times$ at all, X is not an IND and $f_{(\mathfrak{r},\mathfrak{s})}(X) = 0$. ■

Lemmas 3.12 and 3.13 imply that, given an antimonotone, 3-normalized formula φ, we can build an instance $(\mathfrak{r}, \mathfrak{s})$ of Inclusion Dependency$_{\mathit{Identity}}$ in FPT-time (even polynomial) such that $f_\varphi = f_{(\mathfrak{r},\mathfrak{s})}$. Together with the findings of Section 3.3.2, this finishes the proof of Theorem 3.2 showing that Inclusion Dependency$_{\mathit{Identity}}$ and Inclusion Dependency are $\mathbf{W}[3]$-complete.

3.3.4 Discovery

As we did with minimal unique column combinations and functional dependencies, we can lift our results from detecting a single inclusion dependency to

discovering all of them. It turns out that there is a parsimonious equivalence with the enumeration of assignments to antimonotone, 3-normalized formulas, as detailed in Theorem 3.4. The key observations to prove this are once again that the reductions above are polynomial time computable, independently of the parameter, and that they preserve inclusions.

Lemmas 3.12 and 3.13 describe how to turn the formula φ in polynomial time into a pair of databases over the common schema R, which is effectively the same as $\texttt{Var}_\varphi$, such that the inclusion dependencies $X \subseteq R$ are in canonical correspondence with the satisfying assignments $A \subseteq \texttt{Var}_\varphi$. Moreover, the parameterized reduction preserves inclusion relations between the solutions such that the maximal dependencies also correspond to the maximal assignments. In other words, Lemmas 3.12 and 3.13 induce a parsimonious reduction from Enumerate Maximal Satisfying WA3NS Assignments to Discover Maximal $\text{INDs}_{\mathit{Identity}}$. It is also easy to see that Lemma 3.10 implies a parsimonious reduction from the enumeration of such restricted inclusion dependencies to the general Discover Maximal INDs problem. The lemma does nothing else but invalidating all non-identity mappings between the columns. Finally, Lemma 3.11 shows how to translate general inclusion dependencies back to an antimonotone, 3-normalized formula φ. Observe that the (inclusion-wise) maximal satisfying assignments of the resulting formula correspond exactly to the notion of maximality for general INDs (see Sections 2.3.3 and 3.3.1 for details). This shows the equivalence of all the enumeration problems involved. Again, the space complexity is preserved up to polynomial factors by the parsimonious reductions.

We complete the proof of Theorem 3.4 by showing that the problems are at least as hard as Transversal Hypergraph. This is an easy exercise using the structure of antimonotone CNFs.

▶ **Lemma 3.14.** The enumeration of maximal satisfying assignments of antimonotone Boolean formulas in conjunctive normal form is equivalent to Transversal Hypergraph under parsimonious reductions. In particular, Enumerate Maximal Satisfying WA3NS Assignments is at least as hard as the Transversal Hypergraph problem. ◀

Proof. For the reductions, in both directions, we identify the (disjunctive) clauses of an antimonotone CNF formula φ with the sets of variables they contain. To spell it out, let $\texttt{Var}_\varphi$ be the set of all variables of φ and $C_1, \ldots, C_m \subseteq \texttt{Var}_\varphi$ the clauses. Since the formula is antimonotone, any C_i is satisfied iff there is a

variable $x \in C_i$ that is assigned FALSE. In other words, an assignment $A \subseteq \mathtt{Var}_\varphi$ (the set of TRUE variables) is satisfying iff its complement $\overline{A} = \mathtt{Var}_\varphi \setminus A$ is a hitting set of the hypergraph $\{C_i\}_{i\in[m]}$. Assignment A is maximal in that regard iff $\overline{A} \in \mathrm{Tr}(\{C_i\}_i)$ is a minimal transversal. In the very same fashion, we can construct from any hypergraph $(\mathcal{H}, V)$ an antimonotone CNF formula on the variable set $\{x_v \mid v \in V\}$ by setting $\varphi = \bigwedge_{E\in\mathcal{H}} \bigvee_{v\in E} \neg x_v$. Complementing the maximal satisfying assignments of φ recovers the minimal hitting sets of $\mathcal{H}$.

The second part of the lemma follows from any CNF formula being also 3-normalized by viewing literals as conjunctive clauses of length 1. ■

3.4 Conclusion

We determined the complexity of the detection problems for multi-column dependencies parameterized by the solution size. Our results imply that they do *not* admit fixed-parameter tractable algorithms, unless the **W**-hierarchy collapses. This is unfortunate as the choice of parameter is very natural in that the requirement of a small solution size is regularly met in practice. Notwithstanding those barriers, one can still obtain FPT-algorithms by using *other* parameters. For example, to solve the problem UNIQUE COLUMN COMBINATION for a database over some schema R, one can consider all subsets of R and check for each whether it is a unique column combination. (The same considerations also hold for FUNCTIONAL DEPENDENCY and INCLUSION DEPENDENCY.) This takes polynomial time for each of the $2^{|R|}$ subsets. Thus, this leads to an FPT-algorithm with $|R|$ as parameter. This is of course not very satisfying, as assuming $|R|$ to be small is a much stronger assumption than assuming the solution size to be small.

Similarly, one could consider the maximum number d of attributes on which two tuples in a relation r disagree. The standard bounded search tree technique then gives an FPT-algorithm for UNIQUE COLUMN COMBINATION with respect to the sum $d+k$. However, assuming that any two pairs in a relation differ only on a few columns seems to be unrealistic for most data sets. More structural research is needed to identify properties of realistic databases that yield parameters for which the detection problems are tractable. This could involve designing multivariate algorithms with more than one parameter and may even lead to further improvements in the run time of practical methods.

On the other hand, our results regarding the discovery of all dependencies of a certain type in a database are indeed able to explain the good run times

in practice. We proved that the profiling of relational data has the transversal hypergraph problem at its core. Although the exact complexity of the latter is still open, there are many empirically efficient algorithms known. Most importantly, modern algorithms for the enumeration of hitting sets have the advantage that their space complexity is only linear in the input size, this is a feature many data profiling algorithms are still lacking today. In the following chapters, we continue to follow the lead of hitting sets and small solution sizes to develop improved enumeration algorithms.

4 Enumeration in Data Profiling

We devise an enumeration algorithm for minimal hitting sets. It uses linear space and has delay $O(m^{k^*+1}n^2)$*, where* k^* *is the transversal rank. The enumeration uses a search tree pruned by an extension oracle. We prove that the extension problem for minimal hitting sets is* $\boldsymbol{W}[3]$*-complete when parameterized by the size of the set to be extended. We also give several fine-grained lower bounds on its complexity. In an experimental study, we verify that hypergraphs arising from real-world databases are particularly suitable for enumeration.*

4.1 Introduction

Motivated by the results in the previous chapter, we approach the discovery of minimal unique column combinations via the Transversal Hypergraph problem (see Section 3.2.1 for a formal definition). In the absence of an output-polynomial algorithm for general inputs, special classes of hypergraphs have received a lot of attention. For example, it is known that the enumeration of minimal hitting sets is tractable when restricted to hypergraphs with bounded rank, bounded maximum degree [Bor+00], or dual-conformality [Kha+07], as well as for acyclic hypergraphs [EG95; EGM03]. Here, we are interested in the case where the solutions are small. Let k^* denote the *transversal rank* of a given hypergraph, that is, the maximum cardinality of its minimal hitting sets. Indeed, it is very common for hypergraphs arising in data profiling to have low transversal rank, see [Kru+16; Pap+15] and our own analysis in Section 4.5.

Eiter and Gottlob [EG95] gave an output-polynomial algorithm for hypergraphs for which k^* is a constant. Unfortunately, it is not usable for data profiling applications much for the same reason why their equivalence between Transversal Hypergraph and Discover Minimal UCCs is not practical: the *space* consumption of their method scales with the number of solutions. Also, one would like to have a guarantee on the *delay*, the worst-case time between two consecutive outputs, that is independent of the output size. Lastly, although

the transversal rank can be expected to be small usually no a priori bound on k^* is known before the enumeration. In fact, it is **W**[1]-hard to compute k^* and **NP**-hard to approximate [Baz+18]. We are able to improve in all those aspects and obtain an algorithm that is oblivious to k^*, has a space requirement independent of the output, and, in the case that k^* is indeed constant, has polynomial delay.

Central to our algorithm is a subroutine that decides for a set X of vertices whether it is contained in any minimal hitting set. We examine the complexity of the *extension problem*, when parameterized by the cardinality $|X|$ of the partial solution, and prove it as yet another natural problem to be complete for **W**[3]. Similar to the Inclusion Dependency problem already being used by other researchers, Casel et al. [Cas+21] (previously announced in [Cas+18]) employ the techniques we develop here for the extension to minimal hitting sets in order to prove **W**[3]-hardness already for the special case of extension to minimal dominating sets in bipartite graphs.

We also investigate the extension problem with tools from fine-grained complexity. Assuming the Strong Exponential Time Hypothesis (SETH), we prove that our subroutine algorithm is almost optimal. Moreover, we argue that closing the remaining gap between the upper and lower bound is likely to be hard, using a nondetermistic extension of SETH recently conjectured by Carmosino et al. [Car+16]. Next, we give an overview of the results of this chapter in detail.

Main Results. We solve the Transversal Hypergraph problem simultaneously with polynomial delay and space on hypergraphs of bounded transversal rank. More generally, we devise an enumeration algorithm that does not need to know k^*. Its analysis, however, depends on the transversal rank.

▶ **Theorem 4.1.** There exists an algorithm that on n-vertex, m-edge hypergraphs enumerates the minimal hitting sets with delay $O(m^{k^*+1}n^2)$ in $O(mn)$ space, where k^* is the maximum cardinality of any minimal hitting set. ◀

The algorithm employs a tree search pruned by deciding for a given set X of vertices whether it can be extended to a minimal hitting set. We analyze the parameterized complexity of this decision with respect to the parameter $|X|$.

▶ **Theorem 4.2.** The extension problem for minimal hitting sets on a partial solution X is complete for the class **W**[3] when parameterized by $|X|$. ◀

Indeed, we design an efficient enumeration algorithm using a reduction to a hard decision problem. This only makes sense because extension is tractable provided that X contains only a few vertices.

▶ **Theorem 4.3.** There exists an algorithm that decides for an n-vertex, m-edge hypergraph and a set X of vertices whether X is contained in any minimal hitting set in time $O(m^{|X|+1} n)$ and space $O(mn)$. ◀

It is natural to ask whether the exponential dependency on $|X|$ in the running time can be improved. We give several conditional lower bounds, all of which indicate that Theorem 4.3 is close to optimal. They present a trade-off between the strength of the conjecture one is willing to assume and the resulting bound.

▶ **Theorem 4.4.** Let $f\colon \mathbb{N} \to \mathbb{N}$ be an arbitrary computable function. No algorithm can decide for an n-vertex, m-edge hypergraph and a set X of vertices whether X is contained in any minimal hitting set

(*i*) in time $f(|X|) \cdot \mathsf{poly}(m, n)$, unless $\mathsf{W}[3] = \mathsf{FPT}$;

(*ii*) in time $f(|X|) \cdot (m+n)^{o(|X|)}$, unless $\mathsf{W}[2] = \mathsf{FPT}$;

(*iii*) in time $m^{|X|-\varepsilon} \cdot \mathsf{poly}(n)$ for any constant $|X| \geq 2$ and $\varepsilon > 0$, unless the Strong Exponential Time Hypothesis (Hypothesis 2.3) fails. ◀

The SETH-lower bound matches our algorithmic result up to a factor of m. There is a complexity-theoretic obstacle for closing the remaining gap. We argue that if one could show tight SETH-hardness of the extension problem with a time bound of $m^{|X|+1-\varepsilon} \cdot \mathsf{poly}(n)$ via a *deterministic* reduction, this would refute the Nondeterministic Strong Exponential Time Hypothesis (NSETH) [Car+16] and thereby resolve several open problems in circuit complexity and satisfiability.

Finally, we evaluate an implementation of our algorithm by applying it to the Discover Minimal UCCs problem (using the parsimonious reduction from Observation 2.2). Our experiments show that our method is much faster on hypergraphs stemming from real-world databases than the running time bounds would suggest. In practice, a few simple checks can avoid the worst-case behavior on many instances, which boosts the performance. We also confirm the low memory footprint of our approach.

4.2 Enumerating Minimal Hitting Sets

In this section, we outline our enumeration algorithm for minimal hitting sets. Our main motivation comes from data profiling, so we design our method with an eye on instances that have small solutions. Nevertheless, we aim for a general-purpose algorithm and do not restrict the possible input hypergraphs. Therefore,

the algorithm does not make any assumptions on the inputs and relies only on the given hypergraph itself. Its analysis, however, incorporates the transversal rank. Before we present our algorithm, we discuss some alternative approaches and asses how we can improve upon them.

4.2.1 On the Transversal Rank

We review here what is known algorithmically about the transversal rank k^*. This also serves to highlight the subtle differences in measuring the complexity of an enumeration algorithm. Eiter and Gottlob [EG95] showed that the Transversal Hypergraph problem can be solved in incremental polynomial time on instances for which k^* is bounded. Their result hinges on the following proposition.

▶ **Proposition 4.5 (Eiter and Gottlob [EG95]).** Let $\mathcal{H}, \mathcal{G}$ be hypergraphs on the same vertex set V and $k = \operatorname{rank}(\mathcal{G})$. There exists an algorithm that decides whether $\mathcal{G} = \operatorname{Tr}(\mathcal{H})$ in time $O(k|\mathcal{H}||\mathcal{G}| + k(|\mathcal{H}|+|\mathcal{G}|)|V|^k + |\mathcal{H}|^{k+2}|V|)$ and space $O((|\mathcal{H}|+|\mathcal{G}|)|V|)$. Moreover, if $\mathcal{G} \subsetneq \operatorname{Tr}(\mathcal{H})$, the algorithm finds a minimal transversal $T \in \operatorname{Tr}(\mathcal{H})\setminus\mathcal{G}$ within the same bounds. ◀

The enumeration starts with the empty hypergraph $\mathcal{G} = \emptyset$ and repeatedly checks whether $\mathcal{G} = \operatorname{Tr}(\mathcal{H})$, that is, whether $\mathcal{G}$ already contains all solutions. If not, a new solution $T \notin \mathcal{G}$ is computed. Note that $\mathcal{G} \subseteq \operatorname{Tr}(\mathcal{H})$ is an invariant, whence $k = \operatorname{rank}(\mathcal{G})$ is always at most $k^* = \operatorname{rank}(\operatorname{Tr}(\mathcal{H}))$. However, this approach has two drawbacks. It is already unfortunate that the delay depends on $|\mathcal{G}|$ and thus on $|\operatorname{Tr}(\mathcal{H})|$, but it is indeed prohibitive in practice that the space consumption scales with the number of solutions.

If one is working with a class of hypergraphs for which one suspects k^* to be small, albeit no a priori bound is known, one could be tempted to compute the transversal rank[1] first and then brute-force all sets up to that size. Computing k^* is **NP**-hard [Che+90; CN08]. Bazgan et al. [Baz+18] further showed that it is **W**[1]-hard, parameterized by k^*, and that k^* cannot be approximated within a factor of $n^{1-\varepsilon}$ for any constant $\varepsilon > 0$, unless **P** = **NP**.

The parameterized hardness stems from the potentially unbounded size of the hyperedges. Fernau [Fer05] showed that the transversal rank of a graph[2] can

1 Occasionally the term Maximum Minimal Hitting Set is used both for the maximization problem as well as the decision problem whether $k^* \geq k$ for a given integer k [Dam11].

2 This is more commonly known as Maximum Minimal Vertex Cover [BDP15; Fer05].

be computed in FPT-time by presenting an algorithm running in time $O(2^{k^*}) + \mathsf{poly}(n)$, which was later improved to $1.5397^{k^*} \cdot \mathsf{poly}(n)$ [BDP15]. For an arbitrary constant c, Damaschke [Dam11] used Proposition 4.5 to give an algorithm that computes the transversal rank of a hypergraph whose edges have at most c vertices in time $c^{k^*} \cdot p_c(m,n)$, where p_c is a polynomial whose degree depends on c. If c is seen as another parameter (namely, computing k^* parameterized by $c + k^*$), there is an FPT-algorithm running in time $2^{ck^*} \cdot \mathsf{poly}(m,n)$ [Ara+21].

Returning to hypergraphs with unbounded edge size, the parameterized reduction in [Baz+18, Theorem 23] that shows the $\mathsf{W}[1]$-hardness of computing the transversal rank has a quadratic blowup in the parameter. The lower bound on Independent Set by Chen et al. [Che+06] (see Proposition 2.6) thus implies that k^* cannot be computed in time $f(k^*)(m+n)^{o(\sqrt{k^*})}$ for any computable function f, unless the Exponential Time Hypothesis (ETH; Hypothesis 2.4) fails. Very recently and independently of each other Araújo et al. [Ara+21] as well as Dublois, Lampis, and Paschos [DLP21] raised the bound to $f(k^*)(m+n)^{o(k^*)}$. This essentially matches the currently fastest algorithm, which uses the following characterization of the transversal rank by Berge and Duchet [BD75]. Recall from Section 2.2 that a hypergraph is *Sperner* if none of its edges is contained in another, that we can make any hypergraph $\mathcal{H}$ Sperner in quadratic time by computing its *minimization* $\min(\mathcal{H})$, and that $\mathcal{H}$ and $\min(\mathcal{H})$ have the same transversal rank (in fact, the same transversals). For a hypergraph $(V, \mathcal{H})$, subhypergraph $\mathcal{H}' \subseteq \mathcal{H}$, and vertex $v \in V$, let $\deg_{\mathcal{H}'}(v) = |\{E \in \mathcal{H}' \mid v \in E\}|$ denote the *degree of v in $\mathcal{H}'$*.

▶ **Proposition 4.6 (Berge and Duchet [BD75; Ber89]).** Let $(V, \mathcal{H})$ be a Sperner hypergraph and $k \geq 2$ an integer. The transversal rank of $\mathcal{H}$ is at most k if and only if, for all subhypergraphs $\mathcal{H}' \subseteq \mathcal{H}$ with $|\mathcal{H}'| = k+1$ edges, there exists an edge of $\mathcal{H}$ that is contained in the set $\{v \in V \mid \deg_{\mathcal{H}'}(v) > 1\}$. ◀

When computing the transversal rank k^*, the assumption of the input being Sperner does not loose generality. We can test the condition of Proposition 4.6 for increasing k. The value that satisfies it for the first time is k^*. The last iteration dominates the running time, which gives a bound of $O\left(\binom{m}{k^*+1}(k^*+m)\,n\right) = O(m^{k^*+2}\,n)$. The subsequent test of the sets with up to k^* vertices adds another $O(mn^{k^*})$ term. The enumeration time is polynomial for bounded k^*, but the algorithm does not admit any non-trivial guarantees on the delay. Also, the space requirement again depends on the total number of solutions because one has to avoid testing supersets of minimal solutions.

In contrast, we give an algorithm with delay $O(m^{k^*+1} n^2)$, which is better than the bound above for $m > n$. More importantly though, our algorithm uses space that is only linear in the input size, regardless of k^*.

4.2.2 Backtracking Enumeration with an Extension Oracle

It is a common pattern in the design of enumeration algorithms to base them on a so-called *extension oracle* as introduced by Lawler [Law72]. The oracle, tailored to the combinatorial problem at hand, is queried with a set of elements of the underlying universe and decides whether there exists a solution that contains these elements. Applications of this technique usually involve settings in which the extension problem is solvable in polynomial time, like for cycles and spanning trees [RT75], motif search in graphs [Bjö+15], or satisfying assignments for restricted fragments of propositional logic [CH97]. For us, the situation is different in that the extension problem for minimal hitting sets is **NP**-complete [BGH98]. We show later in Section 4.3 that the problem is also hard in a parameterized sense. At first, it may seem paradoxical that reducing enumeration to a hard decision problem can speed up the resulting algorithm. We exploit the fact that the time needed to solve the extension problem is small (enough) for sets that contain only a few vertices.

The original oracle technique [Law72] consists of fixing certain elements of the partial solution and then extending it to the optimum, with respect to a certain ranking function, among all objects that share the fixed elements. During the computation the new candidates are stored in a priority queue. The main bottleneck is the space demand of the queue. For every partial solution, the number of newly introduced candidates can be equal to the size of the universe, meaning exponential growth. Therefore, modifications are necessary to implement the technique efficiently.

In addition to the extension oracle, we use a decision tree to guide the search for minimal solutions in the power lattice of all subsets of the universe. This is known as *backtracking enumeration* [RT75] or *flashlight technique* [MS19]. It allows us to reduce the space requirement to only polynomial in the input.

In the following, we show how to combine both ideas. Let $(V, \mathcal{H})$ be a hypergraph. Suppose we are given an oracle that, queried with disjoint sets $X, Y \subseteq V$, answers whether there exists a minimal solution $T \in \mathrm{Tr}(\mathcal{H})$ such that $X \subseteq T \subseteq V \setminus Y$, that is, whether X is extendable avoiding Y. We use this to enumerate all such T. If $X \cup Y = V$, this can only be $T = X$ itself. Otherwise,

Algorithm 1: Recursive algorithm for Transversal Hypergraph.
The initial call is `enumerate`$(\emptyset, \emptyset, V)$.

Data: Non-empty ordered hypergraph $(V, \leqslant, \mathcal{H})$.
Input: Partition (X, Y, R) of the vertex set V.
Output: The minimal hitting sets $T \in \mathrm{Tr}(\mathcal{H})$ with $X \subseteq H \subseteq V \setminus Y$.

Procedure `enumerate`(X, Y, R):
  if $R = \emptyset$ **then return** X;
  $v \leftarrow \min_{\leqslant} R$;
  isExtendable $\leftarrow$ `extendable`$(X \cup \{v\}, Y)$;
  if *isExtendable* == MINIMAL **then return** $X \cup \{v\}$;
  if *isExtendable* == TRUE **then** `enumerate`$(X \cup \{v\}, Y, R \setminus \{v\})$;
  if (*isExtendable* == FALSE **or** `extendable`$(X, Y \cup \{v\})$ == TRUE) **then**
    `enumerate`$(X, Y \cup \{v\}, R \setminus \{v\})$;

we recursively compute the solutions for the pairs $(X \cup \{v\}, Y)$ and $(X, Y \cup \{v\})$, where v is a vertex neither contained in X nor Y. In other words, we (implicitly) build a binary tree whose nodes are labeled with the pairs (X, Y). The node $(\emptyset, \emptyset)$ is the root and the children of (X, Y) are $(X \cup \{v\}, Y)$ and $(X, Y \cup \{v\})$.

Let $\leqslant$ be a total order on V. Recall from Section 2.2 that this implies a lexicographical order on both the edges of a hypergraph and its transversals. Always choosing the v as the $\leqslant$-smallest element of $V \setminus (X \cup Y)$ gives a universal branching order. This obviates the need of additional communication between the nodes or any shared memory. It is another ingredient to reduce the space demand. In particular, we do not need to record previously found solutions to guide the search. Distinct branches of the tree are independent making the algorithm trivially parallelizable. This is, however, not the focus of this work.

In the absence of any pruning, the recursion would produce the full binary tree with leaves $(X, V \setminus X)$ for *every* possible set $X \in \mathcal{P}(V)$. However, we only need to enter the subtree if one of its leaves is labeled with a minimal hitting set. For the subtree rooted in (X, Y), this is the case iff X can be extended to a minimal hitting set without the vertices in Y.

We formalize this approach in Algorithm 1. Assume for now that subroutine `extendable`(X, Y) solves the extension problem for the given pair of sets and

additionally reports if X itself is already a minimal hitting set. Namely, it returns MINIMAL if $X \in \mathrm{Tr}(\mathcal{H})$, TRUE if there exists some $T \in \mathrm{Tr}(\mathcal{H})$ with $X \subsetneq T \subseteq V \setminus Y$, and FALSE otherwise. We defer the implementation details of `extendable` to Section 4.4. Procedure `enumerate` handles the work inside a node of the decision tree. Besides the depth-first, pre-order traversal of the tree, it also exercises two short-cut evaluations. For this, the ternary variable *isExpendable* holds the result of the first check. If the set $X \cup \{v\}$ is a minimal solution, we output it and return. If it cannot be extended, we immediately recurse on the right-child *without* calling the potentially expensive second check `extendable`$(X, Y \cup \{v\})$. The initial call is `enumerate`$(\emptyset, \emptyset, V)$.

▶ **Lemma 4.7.** Let $(V, \mathcal{H}, \leqslant)$ be a non-empty ordered hypergraph. Suppose, for disjoint sets $X, Y \subseteq V$, subroutine `extendable`(X, Y) decides whether there exists a $T \in \mathrm{Tr}(\mathcal{H})$ with $X \subseteq T \subseteq V \setminus Y$ and, if so, whether $X = T$. Then, Algorithm 1 enumerates the edges of $\mathrm{Tr}(\mathcal{H})$ in $\leqslant$-lexicographical order. ◀

Proof. The correctness is almost immediate from the discussion above. Only the shortcut evaluations have not yet been argued. If the set $X \cup \{v\}$ is not only extendable without Y, but even minimal itself, then adding any more vertices from $V \setminus (X \cup Y \cup \{v\})$ will make it unextendable. Adding these vertices to Y instead does not change $X \cup \{v\}$ being a minimal solution. In summary, we already know in advance the outcomes of all extension checks in the whole subtree rooted in $(X \cup \{v\}, Y)$. The set $X \cup \{v\}$ is the only solution that remains in that tree and we can safely output it and backtrack.

Regarding the second shortcut in line 7, the recursion enters the node (X, Y) only if there exists a $T \in \mathrm{Tr}(\mathcal{H})$ with $X \subsetneq T \subseteq V \setminus Y$. If the first check `extendable`$(X \cup \{v\}, Y)$ returns FALSE, no such T contains the vertex v. Instead, all solutions occur in the subtree rooted at the right child $(X, Y \cup \{v\})$ and we do not need to perform the second evaluation. Note that `extendable`$(X, Y \cup \{v\})$ cannot return MINIMAL due to $X \neq T$.

If $\mathcal{H}$ has no hitting sets (that is, $\emptyset \in \mathcal{H}$) both checks in lines 6 and 7 fail already in the root node and the algorithm returns without an output. Here, we use the assumption that $\mathcal{H}$ has at least one vertex, whence $R = V \neq \emptyset$ holds in the root.

Finally, we prove that the algorithm outputs the minimal transversals in lexicographical order. First, observe that the labeling of the nodes is injective as it encodes the unique path from the root. To see this, let $v_1 \leqslant v_2 \leqslant \ldots \leqslant v_n$ be the total order. Any node with distance k to the root has $X \cup Y = \{v_1, \ldots, v_k\}$ and

X contains exactly those branching nodes at which the recursion entered the left child. Now let $a = (X_a, Y_a)$ and $b = (X_b, Y_b)$ be two distinct leaves such that the pre-order traversal visits a before b. We have $X_a \neq X_b$ from the injective labeling, whence the symmetric difference $X_a \triangle X_b = (X_a \setminus X_b) \cup (X_b \setminus X_a)$ is non-empty. Define $v = \min_{\leqslant} X_a \triangle X_b$. This is the branching vertex of the lowest common ancestor of a and b. Algorithm 1 first tries to add v to the current partial solution in line 6, from which $v \in X_a$ and $X_a \leqslant_{\text{lex}} X_b$ follow. ■

Algorithm 1 bears some similarity to the backtracking method by Elbassioni, Hagen, and Rauf [EHR08, Figure 1]. The main difference is the search for new solutions. In our algorithm, the nodes in the decision tree maintain the partial solution X and additionally the set Y of vertices that have already been excluded. The branching vertex v is chosen, somewhat arbitrarily, by the order $\leqslant$. In contrast, the algorithm in [EHR08] works only on the partial solution X and explicitly computes a new vertex to extend it, which is computationally expensive. Also, their check whether X is already minimal is redundant. This information can be obtained as a by-product of a careful implementation of the subroutine `extendable` at no extra cost, see Lemma 4.17.

We use the order on the vertex set to reduce the need for external coordination. However, the induced lexicographic order on the outputs has applications in data profiling as well. It ensures that interesting unique column combinations are discovered first. Suppose the attributes of a database are ranked by importance, then the lexicographic enumeration starts with those combinations that contain many important attributes. However, the order also raises some complexity-theoretic issues. Johnson, Papadimitriou, and Yannakakis [JPY88] proved that the computation of the lexicographically smallest minimal hitting set is **NP**-hard.[3] Therefore, it is unlikely that any implementation of the extension subroutine can lead to Algorithm 1 having polynomial delay on *all* ordered hypergraphs. Notwithstanding, we present an implementation such that our algorithm achieves polynomial delay at least on instances with bounded transversal rank. We also evaluate the impact of the order on the empirical run time on real-world databases in our experiments in Section 4.5.

3 The **NP**-hardness of computing the lexicographically smallest minimal hitting set does *not* conflict with the statement by Eiter, Gottlob, and Makino [EGM03, Remark 3.1] that the "[lexicographically] smallest" solution can be obtained in linear time. The definition of lexicographic order in [EGM03] is exactly inverse to the one in [JPY88]. We use the latter in this thesis.

4.3 The Minimal Hitting Set Extension Problem

We previously assumed an oracle deciding whether a set of vertices can be extended to a minimal hitting sets. Here, we examine the computational hardness of this decision. The insights gained here will later lead to an algorithm for the subroutine with an almost optimal running time.

It is easy to see that, for a hypergraph $(V, \mathcal{H})$ and disjoint sets $X, Y \subseteq V$, there exists a minimal transversal $T \in \mathrm{Tr}(\mathcal{H})$ such that $X \subseteq T \subseteq V \setminus Y$ if and only if the *truncated* hypergraph $\mathcal{H}' = \{E \setminus Y \mid E \in \mathcal{H}\}$ has a minimal hitting set T with $X \subseteq T$. Indeed, the witnessing transversal T is the same for both $\mathcal{H}$ and $\mathcal{H}'$. We thus define the extension problem as follows.

Minimal Hitting Set Extension (MinHSExt)

Instance: A hypergraph $(V, \mathcal{H})$ and a sets $X \subseteq V$.

Parameter: The cardinality $|X|$.

Decision: Is there a minimal hitting set T of $\mathcal{H}$ with $X \subseteq T$?

Both the classical and parameterized complexity of this problem have received quite some attention. Boros, Gurvich, and Hammer [BGH98] showed that the unparameterized variant of MinHSExt is **NP**-complete. By now, it is known that already many restricted cases are intractable. For example, Mary observed in his PhD thesis [Mar13] (see also [Kan+14]) that extension to minimal dominating sets is **NP**-hard, a result that has subsequently been refined by Bazgan et al. [Baz+18] to planar cubic graphs. In a separate work, Bazgan et al. [Baz+20] showed that extension to minimal vertex covers (that is, MinHSExt in hypergraphs of rank 2) is **NP**-hard as well. However, it was proven in [BGH98] that extension is polynomial-time solvable if the size of the partial solution $|X|$ is constant. This, and the fact that the hitting sets in many applications are small, warrants a parameterized investigation with respect to $|X|$. However, Casel et al. [Cas+19] proved $\mathbf{W}[1]$-completeness for extension to minimal vertex covers, which implies the $\mathbf{W}[1]$-hardness of Minimal Hitting Set Extension.

It is known that extension can be reduced to a certain covering problem in hypergraphs, see [BGH98] and Proposition 4.8 below for a formal definition. We generalize this result by proving that the extension and covering problems are in fact equivalent under parameterized reductions. We then use this equivalence to show that Minimal Hitting Set Extension is another natural problem to

be complete for the parameterized complexity class **W**[3]. We further prove conditional lower bounds on the running time of any algorithm for the extension problem, assuming that certain collapses in the **W**-hierarchy do not occur or that the Strong Exponential Time Hypothesis holds, respectively.

4.3.1 $W[3]$-Completeness

We present necessary and sufficient conditions for a set of vertices to be a subset of some minimal hitting set. This broadens the characterization of minimal transversals in Observation 2.1 in a natural way. The result appears implicitly in [BGH98], we give a self-contained proof here.

▶ **Proposition 4.8 (Boros, Gurvich, and Hammer [BGH98]).** Let $(V, \mathcal{H})$ be a hypergraph and $X \subseteq V$ a set of vertices. There is a $T \in \mathrm{Tr}(\mathcal{H})$ with $X \subseteq T$ if and only if there exists a family of edges $\{E_x\}_{x \in X} \subseteq \mathcal{H}$ such that

(*i*) for every vertex $x \in X$, we have $E_x \cap X = \{x\}$;

(*ii*) for every edge $E \in \mathcal{H}$ contained in $\bigcup_{x \in X} E_x$, we have $E \cap X \neq \emptyset$. ◀

Proof. Let T be a minimal hitting set that contains X. Observation 2.1 guarantees a private edge $E_x \in \mathcal{H}$ with respect to T for every $x \in X$. Let further $E \in \mathcal{H}$ be such that $E \subseteq \bigcup_{x \in X} E_x$. As T is a hitting set, there exists a vertex $y \in E \cap T$. From $(\bigcup_{x \in X} E_x) \cap T = X$, we get $y \in X$. The private edges also fulfill Condition (*ii*).

Conversely, suppose $\{E_x\}_{x \in X}$ is a suitable collection of hyperedges. Condition (*ii*) implies that $H = X \cup (V \setminus \bigcup_{x \in X} E_x)$ is a (not necessarily minimal) hitting set of $\mathcal{H}$. Let $T \subseteq H$ be any *minimal* hitting set, then T contains every $x \in X$ as otherwise E_x would not intersect T by Condition (*i*). ■

We call an edge E a *candidate private edge* for $x \in X$ (with respect to set X) if $E \cap X = \{x\}$ holds. The partial solution X has some extension $T \in \mathrm{Tr}(\mathcal{H})$ iff there is a collection of candidate private edges $\{E_x\}_{x \in X}$ that satisfy Condition (*ii*). Then, the E_x indeed serve as private edges with respect to T.

In light of this characterization, we define an intermediate problem called Multicolored Independent Family. Consider the following task: given k lists of sets together with an additional collection of "forbidden" sets, one has to select one set from each list such that they do not completely cover any forbidden set.

Multicolored Independent Family (MultIndFam)

Instance: A $(k{+}1)$-tuple $(\mathcal{S}_1, \ldots, \mathcal{S}_k, \mathcal{T})$ of hypergraphs on the common vertex set U.

Parameter: The non-negative integer k.

Decision: Are there edges $S_1 \in \mathcal{S}_1, \ldots, S_k \in \mathcal{S}_k$ such that $\bigcup_{i=1}^{k} S_i$ does not contain an edge of $\mathcal{T}$?

The Multicolored Independent Family problem generalizes the **W**[1]-complete Multicolored Independent Set problems on graphs where the vertex set is partitioned into k "color classes" and the desired independent set is required to contain one vertex of each color [Fel+09]. Instead, we now select *sets* of vertices such that their *union* has to be independent. Now the sets have "colors" and the $\mathcal{S}_i$ represent the color classes. Multicolored Independent Set is the special case in which the hypergraphs $\mathcal{S}_i$ consist entirely of singletons and $\mathcal{T}$ of the edges of the graph. Evidently, MultIndFam is **W**[1]-hard.

Next is the FPT-equivalence between Multicolored Independent Family and Minimal Hitting Set Extension. We report the features of the second reduction below in detail as we need them later for the fine-grained lower bounds.

▶ **Lemma 4.9.** Minimal Hitting Set Extension and Multicolored Independent Family are equivalent under polynomial, parameter-preserving reductions.

The reduction from Multicolored Independent Family to the Minimal Hitting Set Extension takes time $O((\sum_{i=1}^{k} |\mathcal{S}_i| + |\mathcal{T}|) \cdot |U|)$ and results in instances with $n = |U| + k$ vertices, $m = \sum_{i=1}^{k} |\mathcal{S}_i| + |\mathcal{T}|$ edges, and parameter $|X| = k$. ◀

Proof. Let $(\mathcal{H}, X)$ be the input to MinHSExt. The set X is extendable iff there are edges $\{E_x\}_{x \in X} \in \mathcal{H}$ with $E_x \cap X = \{x\}$ and their union $\bigcup_{x \in X} E_x$ does not contain any edge that is disjoint from X. This can be phrased as an instance of MultIndFam by defining, for each $x \in X$, the hypergraph $\mathcal{S}_x = \{E \in \mathcal{H} \mid E \cap X = \{x\}\}$. The last hypergraph $\mathcal{T}$ consists of the edges that are disjoint from X. Edges that intersect X in more than one vertex can be cast aside. This is indeed a polynomial reduction.

For the inverse direction, let $(U, \mathcal{S}_1, \ldots, \mathcal{S}_k, \mathcal{T})$ be the instance of Multicolored Independent Family. Let $X = \{x_1, \ldots, x_k\}$ be a set of k new vertices not previously in U. We define the hypergraph $\mathcal{H}$ on the vertex set $V = U \cup X$ by adding all edges of $\mathcal{T}$ as well as $S \cup \{x_i\}$ for every $i \in [k]$ and $S \in \mathcal{S}_i$. Hypergraph

$\mathcal{H}$ can be computed in time

$$O\left(\sum_{i=1}^{k}\sum_{E\in\mathcal{S}_i}|E| + \sum_{E'\in\mathcal{T}}|E'|\right) = O\left(\left(\sum_{i=1}^{k}|\mathcal{S}_i| + |\mathcal{T}|\right)\cdot|U|\right).$$

For any edge $E \subseteq \mathcal{H}$, the set $E \cap V$ is an edge of $\mathcal{S}_i$ if and only if $E \cap X = \{x_i\}$. Moreover, the elements of $\mathcal{T}$ are exactly those edges of $\mathcal{H}$ that are disjoint from X. Therefore, there are $S_1 \in \mathcal{S}_1, \ldots, S_k \in \mathcal{S}_k$ such that $E' \not\subseteq \bigcup_{i=1}^{k} S_i$ holds for all $E' \in \mathcal{T}$ if and only if $\{S_i\}_{i\in[k]}$ satisfies Conditions *(i)* and *(ii)* of Proposition 4.8, that is, if and only if X is extendable to a minimal hitting set of $\mathcal{H}$. ■

The rich structure of Multicolored Independent Family is appreciated when designing algorithms. For the discussion of its complexity, however, it is convenient to also have the freedom to choose the sets from a single list. We thus define the following variant without colors.

Independent Family (IndFam)

Instance: Two hypergraph $\mathcal{S}, \mathcal{T}$ on the common vertex set U and a non-negative integer k.

Parameter: The non-negative integer k.

Decision: Are there k distinct edges $S_1, \ldots, S_k \in \mathcal{S}$ such that $\bigcup_{i=1}^{k} S_i$ does not contain an edge of $\mathcal{T}$?

The two variants are indeed FPT-equivalent.

▶ **Lemma 4.10.** Multicolored Independent Family and Independent Family are equivalent under polynomial, parameter-preserving reductions. ◀

Proof. To reduce MultIndFam to its uncolored variant, it is enough to enforce that selecting two sets of the same color is never a correct solution. They must always cover some forbidden set. Let $(\mathcal{S}_1, \ldots, \mathcal{S}_k, \mathcal{T})$ be an instance of MultIndFam. For every index $i \in [k]$, and $S \in \mathcal{S}_i$, we introduce a new element $x_{S,i}$. The sets are augmented with their respective elements, $S \cup \{x_{S,i}\}$, and the results are collected in the single hypergraph $\mathcal{S}$. Adding the pair $\{x_{S,i}, x_{S',i}\}$ to $\mathcal{T}$ for each i and $S \neq S' \in \mathcal{S}_i$ invalidates all unwanted selections. It is easy to check that this destroys no valid solution.

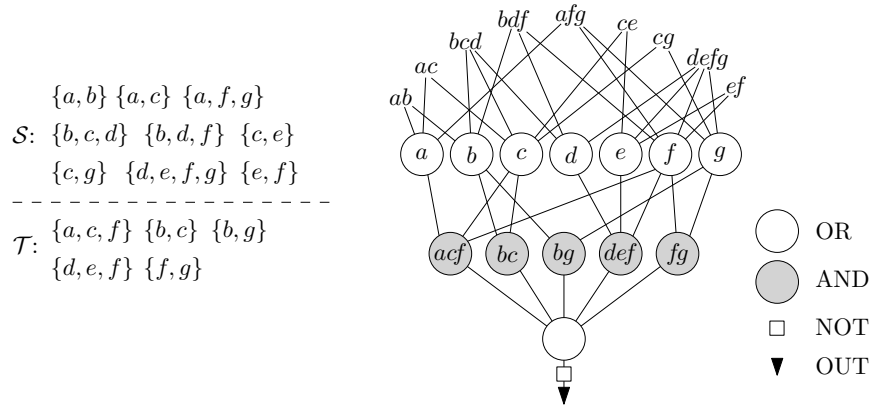


Figure 4.1: Illustration of Lemma 4.11. On the left side is an instance of Independent Family, on the right is the resulting circuit of weft 3. All edges are directed downwards. Selecting $\{a,c\}$, $\{c,e\}$, and $\{c,g\}$ from $\mathcal{S}$ solves the instance for parameter $k = 3$, any other combination of three sets covers a member of $\mathcal{T}$.

For the other direction, we make k copies of $\mathcal{S}$. To ensure that no two copies of the same set are selected together, we take a new element $x_{S,i}$ for each $S \in \mathcal{S}$, $i \in [k]$, define $\mathcal{S}_i = \{S \cup \{x_{S,i}\} \mid S \in \mathcal{S}\}$, and add the sets $\{x_{S,i}, x_{S,j}\}_{i \neq j}$ to $\mathcal{T}$. ■

We now prove that Independent Family is complete for **W**[3]. This transfers to Multicolored Independent Family and Minimal Hitting Set Extension via the reductions in Lemmas 4.9 and 4.10. In Section 3.3.2, we showed the membership of the Inclusion Dependency problem in **W**[3] by reducing it to Weighted Antimonotone 3-normalized Satisfiability. Here, we instead use the more flexible Weighted Circuit Satisfiability problem on weft-3 circuits, which originally was used to define this class. Still, a connection to normalized formulas will become apparent in the hardness proof in Lemma 4.12.

▶ **Lemma 4.11.** There is a polynomial parameter-preserving reduction from Independent Family to Weighted Circuit Satisfiability on constant-depth circuits of weft 3. In particular, Independent Family is in **W**[3]. ◀

Proof. Given an instance $I = (U, \mathcal{S}, \mathcal{T}, k)$ of IndFam, we build a Boolean circuit C of weft 3 that has a satisfying assignment of Hamming weight k iff I is a yes-instance. Figure 4.1 shows an example instance and the resulting circuit. The nodes of C are in one-to-one correspondence to objects in I, slightly abusing

notation we do not distinguish between nodes and their object. The input nodes are the edges of $\mathcal{S}$. Circuit C has a large OR-gate for each vertex in $u \in U$. Node $S \in \mathcal{S}$ is wired to gate u whenever $u \in S$. Next, we introduce a layer of large AND-gates, one for each forbidden set $E \in \mathcal{T}$. Again, u is connected to E iff $u \in E$. All AND-gates lead to a single large OR-gate, its *negated* output serves as the output of the whole circuit.

Note that the circuit can be constructed from instance I in polynomial time. It has depth 4 and weft 3 as every path from an input node to the output passes through exactly one large gate in each if the 3 layers and the (small) NOT-gate. We claim that C is satisfied by setting the input nodes $S_1, \ldots, S_k$ to TRUE if and only if the union $\bigcup_{i=1}^{k} S_i$ contains no edge of $\mathcal{T}$.

Let S_1 to S_k be a selection of k distinct edges of $\mathcal{S}$. Assigning TRUE to the S_i and FALSE to all others satisfies exactly the OR-gates $u \in \bigcup_{i=1}^{k} S_i$. Any AND-gate E of the second layer is satisfied iff all its feeding OR-gates are satisfied, that is, iff $E \subseteq \bigcup_{i=1}^{k} S_i$. The results for all forbidden edges $E \in \mathcal{T}$ are collected by the large OR-gate in the third layer and subsequently negated. Circuit C being satisfied is thus equivalent to *no* edge E being contained in the union of $S_1, \ldots, S_k$. ■

Recall that a formula is antimonotone and 3-normalized if it is a conjunction of subformulas in disjunctive normal form (DNF) with only negative literals. The following formula is an example:

$$((\neg x_1 \wedge \neg x_2 \wedge \neg x_4) \vee (\neg x_3 \wedge \neg x_4)) \wedge ((\neg x_1 \wedge \neg x_3) \vee (\neg x_2 \wedge \neg x_5) \vee (\neg x_1 \wedge \neg x_4 \wedge \neg x_5)).$$

It has satisfying assignments of Hamming weight 0, 1, and 2, but none of larger weight. The intuition behind the **W**[3]-hardness proof is as follows. The circuit C constructed in Lemma 4.11 has a single NOT-gate as the output node. The OR-gates of the first layer are the only ones with fan-out larger than 1, but they are connected exclusively to gates of the second layer. Moving the negation all the way up to the inputs using De Morgan's laws, and duplicating the first layer at most $|\mathcal{T}|$ times hence results in an antimonotone formula that is indeed 3-normalized. We show that this is not a mere artifact of the reduction, but due to a characteristic property of the problem itself. Namely, every such formula can be encoded in an instance of the INDEPENDENT FAMILY problem.

▶ **Lemma 4.12.** There is a polynomial parameter-preserving reduction from WEIGHTED ANTIMONOTONE 3-NORMALIZED SATISFIABILITY to INDEPENDENT FAMILY. In particular, INDEPENDENT FAMILY is hard for **W**[3]. ◀

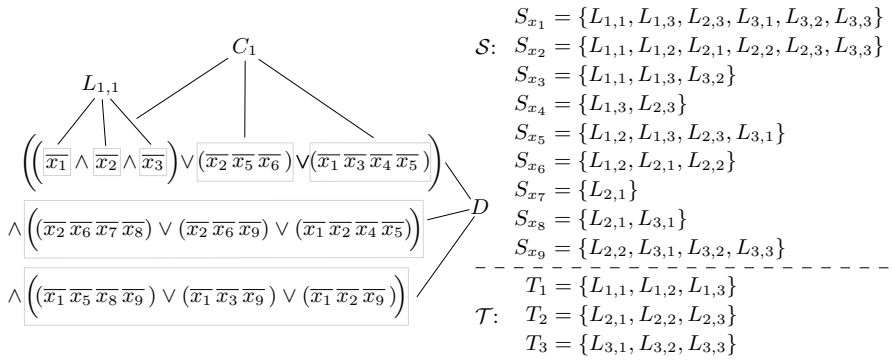


Figure 4.2: Illustration of Lemma 4.12. On the left side is an antimonotone, 3-normalized formula. Negative literals $\neg x_i$ are abbreviated as $\overline{x_i}$ and conjunctions inside a clause as juxtaposition. On the right is the resulting instance of Independent Family. Positions marked with grey boxes are indexed by the respective sets: D for the DNF subformulas, C_1 for the conjunctive clauses of the first subformula, and $L_{1,1}$ for the first clause of the first subformula. The formula admits a satisfying assignment of weight 4 by setting x_4, x_5, x_7, and x_8 to true. Equivalently, the union of the sets S_{x_4}, S_{x_5}, S_{x_7}, and S_{x_8} does not cover any forbidden set in $\mathcal{T}$. No assignment of Hamming weight at least 5 is satisfying.

Proof. A Boolean formula φ on the variable set $\texttt{Var}_\varphi$ is antimonotone and 3-normalized if and only if it can be written as

$$\varphi = \bigwedge_{d \in D} \bigvee_{c \in C_d} \bigwedge_{\ell \in L_{d,c}} \neg x_{d,c,\ell},$$

for an index hierarchy D, $\{C_d\}_{d \in D}$, $\{L_{d,c}\}_{d \in D,\, c \in C_d}$, and $x_{d,c,\ell} \in \texttt{Var}_\varphi$. The index d ranges over the constituent DNF subformulas, c over their conjunctive clauses, and ℓ over the negative literals. Of course, a variable may appear multiple times in the formula, so different triples (d, c, ℓ) may point to the same variable.

We construct an instance $(U, \mathcal{S}, \mathcal{T}, k)$ of Multicolored Independent Family that is a yes-instance if and only if φ has a weight-k satisfying assignment. This is illustrated in Figure 4.2. We take as vertex set the conjunctive clauses $U = \{L_{d,c} \mid d \in D,\, c \in C_d\}$ and add the edge $S_x = \{L_{d,c} \mid \exists \ell \colon x_{d,c,\ell} = x\}$ to $\mathcal{S}$ for each variable $x \in \texttt{Var}_\varphi$. Namely, S_x contains all clauses in which x occurs. The DNF subformulas are represented in the hypergraph $\mathcal{T}$ via the edges $E_d = \{L_{d,c} \mid c \in C_d\}$ for all $d \in D$.

The key observation is the following. Consider a truth assignment represented by the set $A \subseteq \mathtt{Var}_\varphi$ of the variables assigned TRUE. Since φ is antimonotone, clause $L_{d,c}$ is satisfied if and only if *none* of its variables $x_{d,c,\ell}$ is in A. As a result, subformula d is TRUE iff A is *not* a hitting set for the clauses of d.

Suppose the assignment $A = \{x_1, \ldots, x_k\}$ is satisfying. Then, the union $\bigcup_{i=1}^{k} S_{x_i}$ contains exactly the conjunctive clauses that are not satisfied. If this union were to cover any forbidden edge in $\mathcal{T}$, the corresponding subformula, and hence φ, would be unsatisfied, a contradiction. Therefore, $(U, \mathcal{S}, \mathcal{T}, k)$ is a yes-instance of Independent Family. Conversely, let S_{x_1} through S_{x_k} be a selection of edges from $\mathcal{S}$ such that their union covers no member of $\mathcal{T}$. In other words, each subformula has at least one clause that is disjoint from $\{x_1, \ldots, x_k\}$. Assigning TRUE to (exactly) those variables k-satisfies φ. ■

4.3.2 Fine-Grained Lower Bounds

We now discuss consequences of our reductions beyond parameterized complexity. They allow us to derive lower bounds on the running time of any algorithm for Minimal Hitting Set Extension from certain hypotheses. These lower bounds are conditional in that the hypotheses, albeit plausible, are still unproven.

The common belief that **NP**-hard problems do not have polynomial algorithms, or that **W**[1]-hard ones do not admit FPT-solutions, can be cast as *conditional* running time lower bounds in that they rely on unproven hypotheses. In the last decade, this perspective has been developed into the area of fine-grained complexity, see the surveys by Bringmann [Bri19] and Vassilevska Williams [Vas19]. The field tries to determine the exact exponent of the time needed to solve various problems in the polynomial, exponential, and parameterized domain. The proven lower bounds often match closely with the best known algorithms, but they come with the caveat of relying on even more hardness assumptions. Such bounds need to strike a balance between the plausibility of the conjecture and the strength of the result following from it.

We offer three lower bounds on the extension problem. They are presented in order of increasing strength and are respectively derived from ever stronger conjectures about the **W**-hierarchy and Boolean satisfiability. The first one immediately follows from Minimal Hitting Set Extension being **W**[3]-complete. If **W**[3] ≠ **FPT**, there is no FPT-algorithm solving the extension problem in time $f(|X|) \cdot \mathrm{O}((m+n)^c)$ on hypergraphs with n vertices and m edges for any computable f and constant c. Note that the parameterized reductions above also show

that Multicolored Independent Family and Independent Family cannot be solved in time $f(k) \cdot \mathsf{poly}(|\mathcal{S}_1|, \ldots, |\mathcal{S}_k|, |\mathcal{T}|, |U|)$ and $f(k) \cdot \mathsf{poly}(|\mathcal{S}|, |\mathcal{T}|, |U|)$.

The next lower bound follows from the stronger assumption that $\mathsf{W}[2] \neq \mathsf{FPT}$. For this, we use a proposition[4] by Chen et al. [Che+06].

▶ **Proposition 4.13 (Chen et al. [Che+06]).** Let f be an arbitrary computable function. If there exists an algorithm solving the Weighted Antimonotone 3-normalized Satisfiability problem on formulas of size m with n variables in time $f(k)\, n^{o(k)}\, \mathsf{poly}(m)$, then $\mathsf{W}[2] = \mathsf{FPT}$. ◀

Note that the reductions from WA3NS to IndFam, and further to MultIndFam and MinHSExt in Lemmas 4.9, 4.10 and 4.12 are all polynomial-time computable and linear the sense that they increase the parameter by at most a constant factor. In fact, they even preserve the parameter exactly. Any algorithm solving the Minimal Hitting Set Extension problem in time $f(|X|)\,(m+n)^{o(|X|)}$ on n-vertex, m-edge hypergraphs would thus give a fast algorithm for Weighted Antimonotone 3-normalized Satisfiability and thus imply the collapse $\mathsf{W}[2] = \mathsf{FPT}$. Similar bounds also hold for the intermediate problems.

The above bound states that the exponent of the worst-case running time for MinHSExt necessarily has a linear dependence on the parameter $|X|$. We show next that the leading coefficient of that dependency is likely to be 1. The Orthogonal Vectors (OV) problem serves as an illustration of this kind of result. Recall that for this problem we are given two sets, each with n binary vectors in d dimensions, and we ought to decide whether there is one vector from each set such that their inner product is 0. Straightforwardly testing all pairs yields an $O(n^2 d)$-time algorithm. Maybe surprisingly, Williams [Wil05] showed that this cannot be improved to $n^{2-\varepsilon} \cdot \mathsf{poly}(d)$ for any constant $\varepsilon > 0$, at least not if one believes that CNF SAT on formulas with n variables cannot be solved in time $O(2^{(1-\varepsilon/2)n})$. Such an improved algorithm would be a huge breakthrough in satisfiability, its conjectured non-existence is the core of the Strong Exponential Time Hypothesis.

We derive our hardness result from a generalization of OV, known as the k-Orthogonal Vectors problem. Let $k \geq 2$ be an integer.

4 The proposition follows from a more general result [Che+06, Theorem 4.2] on the weighted satisfiability of what the authors call structured Π_t-circuits. For $t = 3$, the structure coincides with that of antimonotone, 3-normalized formulas.

k-Orthogonal Vectors (k-OV)

Instance: Sets $A_1, \ldots, A_k \subseteq \{0,1\}^d$ with $|A_1| = \cdots = |A_k| = n$.

Decision: Are there vectors $\vec{x}^{(1)} \in A_1$, $\vec{x}^{(2)} \in A_2$, $\ldots$, $\vec{x}^{(k)} \in A_k$ such that $\sum_{j=1}^{d} \prod_{i=1}^{k} x_j^{(i)} = 0$?

The addition and multiplication are those in $\mathbb{N}$, not the field $\mathbb{F}_2$. We also emphasize that this defines a family of problems, one for each $k \geq 2$, as opposed to a single parameterized problem.

The following conjecture generalizes Hypothesis 2.5 from OV to k-OV.

▶ **Hypothesis 4.14 (k-Orthogonal Vectors conjecture in moderate dimensions (k-OV conjecture)).** For any constants $\varepsilon > 0$ and $k \geq 2$, the k-Orthogonal Vectors problem cannot be solved in time $n^{k-\varepsilon} \cdot \mathsf{poly}(d)$. ◀

It is well-known that a slight change of the reduction in [Wil05] proves that SETH implies Hypothesis 4.14. Nevertheless, it is consistent with our current knowledge that the k-OV conjecture holds while SETH is false. The assumptions $\mathsf{W}[3] \neq \mathsf{FPT}$ and $\mathsf{W}[2] \neq \mathsf{FPT}$ used above also follow from SETH but are possibly much weaker, see the discussion in [Che+06; Cyg+15; IP01; IPZ01]. Again, no inverse connection nor any relation between the conjectures on the **W**-hierarchy and on k-Orthogonal Vectors are known.

We aim to disprove the existence of an algorithm for Minimal Hitting Set Extension running in time $m^{|X|-\varepsilon} \cdot \mathsf{poly}(n)$ for any constant $\varepsilon > 0$ and constant parameter $|X|$. By Lemma 4.9, such an algorithm implies Multicolored Independent Family being solvable in time $(\sum_{i=1}^{k} |\mathcal{S}_i| + |\mathcal{T}|)^{k-\varepsilon} \cdot \mathsf{poly}(|U|)$. We show that the latter assertion contradicts the k-Orthogonal Vectors conjecture.

▶ **Lemma 4.15.** If there exists an algorithm solving Multicolored Independent Family in time $(\sum_{i=1}^{k} |\mathcal{S}_i| + |\mathcal{T}|)^{k-\varepsilon} \cdot \mathsf{poly}(|U|)$ for any constants $\varepsilon > 0$ and $k \geq 2$, then the k-OV conjecture fails. ◀

Proof. Naturally, we reduce from k-OV. Figure 4.3 illustrates the construction. Let $A_1, \ldots, A_k \subseteq \{0,1\}^d$ be sets with n binary vectors each. The constructed instance of MultIndFam has $U = [k] \times [d]$ as its vertex set. Let $\mathbb{1}(\vec{x}) = \{j \in [d] \mid x_j = 1\}$ be the set with characteristic vector $\vec{x}$. For each $i \in [k]$, we let the hypergraph $\mathcal{S}_i$ represent the set A_i by adding the edge $\{i\} \times \mathbb{1}(\vec{x}) = \{(i,j) \mid j \in \mathbb{1}(\vec{x})\}$ for each $\vec{x} \in A_i$. Hypergraph $\mathcal{T}$ contains the edge $F_j = [k] \times \{j\}$ for every $j \in [d]$.

$$
\begin{aligned}
A_1 = \{&111110,\\
&001111,\\
&011011,\\
&010101\}
\end{aligned}
\qquad
\begin{aligned}
\mathcal{S}_1 = \{&\{(1,1),(1,2),(1,3),(1,4),(1,5)\},\\
&\{(1,3),(1,4),(1,5),(1,6)\},\\
&\{(1,2),(1,3),(1,5),(1,6)\},\\
&\{(1,2),(1,4),(1,6)\}\}
\end{aligned}
$$

$$
\begin{aligned}
A_2 = \{&010111,\\
&101110,\\
&011101,\\
&111011\}
\end{aligned}
\qquad
\begin{aligned}
\mathcal{S}_2 = \{&\{(2,2),(2,4),(2,5),(2,6)\},\\
&\{(2,1),(2,3),(2,4),(2,5)\},\\
&\{(2,2),(2,3),(2,4),(2,6)\},\\
&\{(2,1),(2,2),(2,3),(2,5),(2,6)\}\}
\end{aligned}
$$

$$
\begin{aligned}
A_3 = \{&011011,\\
&110110,\\
&011100,\\
&101111\}
\end{aligned}
\qquad
\begin{aligned}
\mathcal{S}_3 = \{&\{(3,2),(3,3),(3,5),(3,6)\},\\
&\{(3,1),(3,2),(3,4),(3,5)\},\\
&\{(3,2),(3,3),(3,4)\},\\
&\{(3,1),(3,3),(3,4),(3,5),(3,6)\}\}
\end{aligned}
$$

$$
\begin{aligned}
\mathcal{T} = \{&\{(1,1),(2,1),(3,1)\},\{(1,2),(2,2),(3,2)\},\\
&\{(1,3),(2,3),(3,3)\},\{(1,4),(2,4),(3,4)\},\\
&\{(1,5),(2,5),(3,5)\},\{(1,6),(2,6),(3,6)\}\}
\end{aligned}
$$

Figure 4.3: Illustration of Lemma 4.15 for $k = 3$. On the left side is an 3-Orthogonal Vectors instance with $n = 4$ vectors in $d = 6$ dimensions. On the right is the resulting instance of Multicolored Independent Family. The three vectors $010101 \in A_1$, $101110 \in A_2$, and $011011 \in A_3$ together are orthogonal, the union of the corresponding edges from $\mathcal{S}_1$, $\mathcal{S}_2$, and $\mathcal{S}_3$ does not contain any edge of $\mathcal{T}$.

Intuitively, F_j being completely covered by the union of hyperedges means that the corresponding vectors all share a 1 in the j-th component.

Let $\vec{x}^{(1)} \in A_1$, $\vec{x}^{(2)} \in A_2$, …, $\vec{x}^{(k)} \in A_k$ be a selection of vectors. For any $i \in [k]$ and $j \in [d]$, we have $x_j^{(i)} = 0$ if and only if (i, j) is *not* contained in the edge $\{i\} \times \mathbb{1}(\vec{x}^{(i)}) \in \mathcal{S}_i$. Moreover, no edge of any other $\mathcal{S}_\ell$, $\ell \neq i$, can contain (i, j). Therefore, we have $F_j \notin \bigcup_{i=1}^{k}(\{i\} \times \mathbb{1}(\vec{x}^{(i)}))$ iff $\prod_{i=1}^{k} x_j^{(i)} = 0$. Finally, this is the case for all $F_j \in \mathcal{T}$ iff the vectors $\vec{x}^{(1)}, \ldots, \vec{x}^{(k)}$ are orthogonal.

Recall that $k \geq 2$ is a constant. There are $\sum_{i=1}^{k} |\mathcal{S}_i| + |\mathcal{T}| = kn + d = O(n+d)$ edges on $|U| = kd$ vertices and the new instance is computable in time $O(knd + kd) = O(nd)$. The algorithm for Multicolored Independent Family running in time $(\sum_{i=1}^{k} |\mathcal{S}_i| + |\mathcal{T}|)^{k-\varepsilon} \cdot \mathsf{poly}(|U|)$ would solve k-Orthogonal Vectors in

$$O((n+d)^{k-\varepsilon}) \cdot \mathsf{poly}(d) = O(n^{k-\varepsilon} + d^{k-\varepsilon}) \cdot \mathsf{poly}(d) = n^{k-\varepsilon} \cdot \mathsf{poly}(d). \qquad \blacksquare$$

4.3.3 The Nondeterministic Strong Exponential Time Hypothesis

Any algorithm solving the Minimal Hitting Set Extension problem in time $m^{|X|-\varepsilon} \cdot \mathsf{poly}(n)$ for arbitrary constants $|X|$ and $\varepsilon > 0$ violates SETH. In Section 4.4, we present an $O(m^{|X|+1}\, n)$-time solution. This raises the question what is the "true" exponent of m. Although we believe that our algorithm is optimal with respect to m, at least up to subpolynomial factors, we sketch an argument why it might be hard to raise the lower bound of Lemma 4.15 to, say, $m^{|X|+1-o(1)} \cdot \mathsf{poly}(n)$.

Carmosino et al. [Car+16] identified a fundamental obstacle for proving SETH-hardness. A co-nondeterministic algorithm for some decision problem is one whose computation path may have nondeterministic transitions. On a yes-instance, every path is required to produce the answer true, on a no-instance, there must be at least one path resulting in false. The only known co-nondeterministic algorithm for CNF SAT that improves over brute force is randomized [Wil16]. The *Nondeterministic Strong Exponential Time Hypothesis* (NSETH) conjectures that this behavior is inherent to the problem in that no non-randomized co-nondeterministic algorithm can break the 2^n-barrier on formulas with n-variables.

▶ **Hypothesis 4.16 (Nondeterministic Strong Exponential Time Hypothesis, NSETH [Car+16]).** For every constant $\varepsilon > 0$, there exists a positive integer k such that no co-nondeterministic algorithm without access to randomness can decide k-CNF SAT on n-variable formulas in time $O(2^{(1-\varepsilon)n})$. ◀

NSETH can be seen as a common generalization of SETH and $\mathsf{NP} \neq \mathrm{co}\mathsf{NP}$. The value of the conjecture lies not so much in its plausibility–it is false for randomized algorithms–but the fact that both proving and refuting NSETH has interesting consequences. Finding a fast co-nondeterministic algorithm for satisfiability would immediately yield new circuit lower bounds, see [Car+16]. Proving NSETH would, among other things, resolve the P vs. NP problem.

The conjecture also rules out the existence of certain fine-grained reductions. Consider a decision problem Π that admits an algorithm $\mathcal{A}$ running in time $T(m, n)$ and also a non-randomized co-nondeterministic algorithm $\mathcal{B}$ running in time $T(m, n)^{1-\varepsilon}$ for some constant $\varepsilon > 0$. If NSETH is true, then no deterministic reduction from CNF SAT to Π can prove that algorithm $\mathcal{A}$ is optimal under SETH since the very same reduction would give an improved co-nondeterministic algorithm for CNF SAT using algorithm $\mathcal{B}$.

For the further discussion regarding the hardness of Minimal Hitting Set Extension, we use the language of first-order model checking. For an introduction to first-order logic in parameterized complexity, see the textbook Flum and Grohe [FG06]. The equivalent Multicolored Independent Family problem can be seen as deciding whether the input $(U, \mathcal{S}_1, \ldots, \mathcal{S}_k, \mathcal{T})$ is a model[5] for

$$\varphi \;=\; \exists x_1 \in \mathcal{S}_1 \ldots \exists x_k \in \mathcal{S}_k \; \forall y \in \mathcal{T} \; \exists z \in U : \; z \in y \wedge \bigwedge_{i=1}^{k} z \notin x_i.$$

MultIndFam is a *graph problem* in the sense that the maximum arity of any relation in $(U, \mathcal{S}_1, \ldots, \mathcal{S}_k, \mathcal{T})$ is 2. Formula φ has k existential quantifiers, followed by a universal one, and then another existential quantifier. We abbreviate this to $\exists^k \forall \exists$. Since MultIndFam is $\mathbf{W}[3]$-complete, the quantifier structure is a characteristic property of the problem, see [FG06].

Let k be a positive integer and let ℓ denote the total number of "edges", meaning the tuples in the binary relations. Note that for MultIndFam ℓ can be as large as $(\sum_{i=1}^{k} |\mathcal{S}_i| + |\mathcal{T}|) \cdot |U|$. Along the lines sketched above, Carmosino et al. [Car+16, Theorem 4] showed that under NSETH the *only* graph problems with $k+2$ quantifiers that can be proven to be SETH-hard with a time bound $\ell^{k+1-o(1)}$ via deterministic reductions are those with quantifier structure $\exists^{k+1}\forall$ or $\forall^{k+1}\exists$.

Using SETH to disprove the existence of an algorithm for Multicolored Independent Family running in time $O(\ell^{k+1-\varepsilon})$, that is,

$$O\left(\left(\left(\sum_{i=1}^{k} |\mathcal{S}_i| + |\mathcal{T}|\right)|U|\right)^{k+1-\varepsilon}\right) = \left(\sum_{i=1}^{k} |\mathcal{S}_i| + |\mathcal{T}|\right)^{k+1-\varepsilon} \cdot \mathbf{poly}(|U|),$$

for any $\varepsilon > 0$, would therefore need to introduce randomness in a non-trivial way or provide a breakthrough co-nondeterministic algorithm for CNF SAT.

4.4 An Algorithm for the Extension Problem

To finish the description of our hitting set enumeration algorithm, we need to implement the subroutine for the extension problem. We not only assumed that

5 Strictly speaking, we express the instance $(U, \mathcal{S}_1, \ldots, \mathcal{S}_k, \mathcal{T})$ as a relational structure over the universe $U \cup \mathcal{S}_1 \cup \cdots \cup \mathcal{S}_k \cup \mathcal{T}$ with unary relations $S_1, \ldots, S_k, T$, where $S_i\, x$ is interpreted as x being an edge of $\mathcal{S}_i$, and one binary relation $\in \subseteq U \times (\mathcal{S}_1 \cup \cdots \cup \mathcal{S}_k \cup \mathcal{T})$.

Algorithm 2: Algorithm for Minimal Hitting Set Extension.

Input: Hypergraph $(V, \mathcal{H})$, $\mathcal{H} \neq \emptyset$, and
disjoint sets $X = \{x_1, \ldots, x_{|X|}\}$, $Y \subseteq V$.
Output: minimal if $X \in \mathrm{Tr}(\mathcal{H})$, true if there is a $T \in \mathrm{Tr}(\mathcal{H})$
with $X \subsetneq T \subseteq V \setminus Y$, and false otherwise.

```
1  if X = ∅ then
2      if V\Y is a hitting set then return TRUE;
3      else return FALSE;
4  initialize hypergraph T = ∅;
5  foreach x ∈ X do initialize hypergraph S_x = ∅;
6  foreach E ∈ H do
7      if E ∩ X = {x} then add E\Y to S_x;
8      if E ∩ X = ∅ then add E\Y to T;
9  if ∃x ∈ X: S_x = ∅ then return FALSE;
10 if T = ∅ then return MINIMAL;
11 foreach (E_{x_1}, ..., E_{x_|X|}) ∈ S_{x_1} × ··· × S_{x_|X|} do
12     W ← ⋃_{i=1}^{|X|} E_{x_i};
13     if ∀T ∈ T: T ⊄ W then return TRUE;
14 return FALSE;
```

we can decide for disjoint sets X and Y whether X can be extended to a minimal hitting set avoiding Y, we additionally claimed that it is possible to find out whether X is itself a solution at no additional cost.

Despite the hardness results, the investigation in Section 4.3 also revealed some structure of the Minimal Hitting Set Extension problem that can be exploited algorithmically. Justified by Lemma 4.9, we approach it via Multi-colored Independent Family. Let $\mathcal{H}$ be the input hypergraph. If $\mathcal{H} = \emptyset$ does not contain a single edge, X is a minimal transversal if and only if $X = \emptyset$ is empty as well. In the remainder we assume that $\mathcal{H}$ is non-empty and solve the extension problem with Algorithm 2. To handle the set Y of excluded vertices, the algorithm computes the truncated hypergraph $\{E \setminus Y\}_{E \in \mathcal{H}}$ and then reduces it to an instance of MultIndFam. In fact, both steps can be computed in one pass (line 4–8).

Lemma 4.15 suggests that we cannot improve much over brute force when solving the resulting instance, at least not in the worst case. There are, however, several sanity checks possible that may avoid unnecessary computations in practice. The first check is the special case of an empty set $X = \emptyset$. It is extendable without using Y if and only if $V \setminus Y$ is a hitting set, that is, iff Y does not contain an edge. The other two checks (in lines 9 and 10) assess whether the instance at hand can be decided immediately. If the checks are inconclusive, the instance is indeed solved by brute force (line 11–14). Note that the existence of a minimal extension is decided without explicitly computing one. As shown in Section 4.2, this is enough for the enumeration.

Recall that $n = |V|$ denotes the number of vertices and $m = |\mathcal{H}|$ the number of edges of the hypergraph. We now show that the running time of Algorithm 2 matches the OV-lower bound of Lemma 4.15 up to an $O(m)$-factor.

▶ **Lemma 4.17.** Let $(V, \mathcal{H})$ be a non-empty hypergraph and $X, Y \subseteq V$ disjoint sets of vertices. Algorithm 2 returns MINIMAL if $X \in \mathrm{Tr}(\mathcal{H})$ is a minimal hitting set, TRUE if there is a $T \in \mathrm{Tr}(\mathcal{H})$ with $X \subsetneq T \subseteq V \setminus Y$, and FALSE otherwise. The algorithms runs in $O((\frac{m}{|X|})^{|X|} \cdot mn)$ time and $O(mn)$ space. ◀

Proof. The first part up to line 10 of the algorithm computes the reduction from MinHSExt to MultIndFam (Lemma 4.9) for the truncated hypergraph $(V \setminus Y, \{E \setminus Y\}_{E \in \mathcal{H}})$. The sanity checks in lines 1, 9 and 10 filter out trivial instances. The foreach-loop starting in line 11 is then brute-forcing the result of the reduction, checking all tuples in the Cartesian product $\prod_{x \in X} \mathcal{S}_x$.

Since $\mathcal{H}$ is non-empty, the empty set $X = \emptyset$ cannot be a hitting set of $\mathcal{H}$. For some $X \neq \emptyset$ to be a hitting set, the corresponding hypergraph $\mathcal{T}$ must be empty, as verified in line 10. Observe that this reduces Proposition 4.8 about candidate private edges to Observation 2.1. Therefore, such an X is minimal iff every $x \in X$ has a private edge, which is exactly what is tested in line 9. Algorithm 2 correctly identifies the minimal transversals X and returns MINIMAL in line 10.

Regarding the time complexity, we assume that all set operations (membership, product, union, intersection, and difference) are implemented such that they take time proportional to the total number of elements contained in the input and output of the operation. Checking whether $V \setminus Y$ is a hitting set and computing the hypergraphs $\mathcal{S}_{x_1}, \ldots, \mathcal{S}_{x_{|X|}}$, and $\mathcal{T}$ can thus be done in time $O(mn)$. The running time is dominated by the brute-force phase. The cardinality of the Cartesian product is maximum if all hypergraphs have the same number of

sets and no edge is cast aside. There are thus at most $(m/|X|)^{|X|}$ many tuples. For each of them, the algorithm computes the union W in $O(|X|n)$ time and checks all forbidden sets in $\mathcal{T}$ in $O(mn)$. The fact that every element of X has a candidate private edge implies $|X| \leq m$ and $O(|X|n + mn) = O(mn)$.

Regarding the space requirement, note that $\mathcal{S}_x$ and $\mathcal{T}$ are all disjoint subhypergraphs of $\{E \setminus Y\}_{E \in \mathcal{H}}$, using at most as much space as $(V, \mathcal{H})$. ■

Finally, we use Lemma 4.17 to prove a guarantee on the maximum delay between consecutive outputs of Algorithm 1. The bound is stated in terms of the transversal rank $k^* = \text{rank}(\text{Tr}(\mathcal{H}))$. Recall that k^* is *not* known to the algorithm, the input consists only of the hypergraph itself. For bounded transversal rank, we achieve polynomial delay. In particular, Algorithm 1 then solves the Transversal Hypergraph problem in output-polynomial time.

▶ **Lemma 4.18.** Consider Algorithm 1 with Algorithm 2 implementing the subroutine `extendable`. On input $(V, \leqslant, \mathcal{H})$, it enumerates the edges of $\text{Tr}(\mathcal{H})$ in $\leqslant$-lexicographical order with delay $O(m^{k^*+1} n^2)$, where $k^* = \text{rank}(\text{Tr}(\mathcal{H}))$. The algorithm uses $O(mn)$ space. ◀

Proof. The correctness was treated in Lemmas 4.7 and 4.17. We have also shown there that the label of the current node contains all relevant information to govern the tree search. In particular, it encodes the path to the node in the (only implicitly constructed) recursion tree for backtracking. The total space usage is thus dominated by the $O(mn)$ of Algorithm 2.

We are left to bound the delay. The height of the tree is $|V| = n$. After exiting a leaf, the pre-order traversal expands at most $2n - 1$ inner nodes before arriving at the next leaf. In the worst case, method `extendable` is invoked in each of them, even with the shortcut evaluations. The $O((\frac{m}{|X|})^{|X|} mn) = O(m^{|X|+1} n)$ subroutine dominates the time spent in each node.

We prove that during the enumeration any set X appearing as the first argument of `extendable` is of cardinality at most $|X| \leq k^*$. To reach a contradiction, assume a node $(X, Y, R = V \setminus (X \cup Y))$ with $|X| > k^*$ is expanded by Algorithm 1. This cannot be the root as X is non-empty. Thus, prior to entering $(X, Y; R)$, either `extendable(X,Y)` has been called or the shortcut evaluation inferred the outcome TRUE from the previous calls. Set X is neither a minimal solution nor can it be extended to one as its cardinality is larger than the transversal rank. The check returned FALSE and (X, Y, R) is never entered, a contradiction. Therefore, the delay is bounded by $(2n - 1) \cdot O(m^{k^*+1} n) = O(m^{k^*+1} n^2)$. ■

4.5 Enumerating Unique Column Combinations

We apply our enumeration algorithm to hypergraphs arising in data profiling as a proof of concept. Specifically, we solve Discover Minimal UCCs. The folklore reduction translates this to the Transversal Hypergraph problem, and we showed in Chapter 3 that encountering hitting sets is really unavoidable when dealing with UCCs. Observation 2.2 implies a two-phased approach for their discovery. First, generate the hypergraph of minimal difference sets. Secondly, list its minimal transversals. The first phase takes time polynomial in the size of the database. The second phase, which has exponential complexity in the worst case, is the focus of this section. In the following, we thus assume that the Sperner hypergraph of minimal difference sets is given as the input.

4.5.1 Data and Experimental Setup

We evaluate our enumeration algorithm on a total of 12 databases. Ten of them are publicly available. These are the `abalone`, `echocardiogram`, `hepatitis`, and `horse` datasets from the University of California Irvine (UCI) Machine Learning Repository;[6] `uniprot` from the Universal Protein Resource;[7] the datasets `civil_service`,[8] `ncvoter_allc`[9] and `flight_1k`[10] provided by the respective authorities of the City of New York, the state of North Carolina, and the federal government of the United States; `call_a_bike` of the German railway company Deutsche Bahn,[11] as well as `amalgam1` from the Database Lab of the University of Toronto.[12] They are complemented by two randomly generated datasets `fd_reduced_15` and `fd_reduced_30` using the *dbtesma* data generator.[13] Databases with more than 100k rows are cut by choosing 100k rows uniformly at random.

The algorithms are implemented in C++ and run on a Ubuntu 16.04 machine with two Intel® Xeon® E5-2690 v3 2.60 GHz CPUs and 256 GB RAM We made

6 archive.ics.uci.edu/ml
7 uniprot.org
8 opendata.cityofnewyork.us
9 ncsbe.gov
10 transtats.bts.gov
11 data.deutschebahn.com
12 dblab.cs.toronto.edu/~miller/amalgam
13 sourceforge.net/projects/dbtesma

Dataset	Columns	Rows	n	m	k^*	UCCs
call_a_bike	17	100 000	13	6	4	23
abalone	9	4177	9	30	6	29
echocardiogram	13	132	12	30	5	72
civil_service	20	100 000	14	19	7	81
horse	29	300	25	39	11	253
uniprot	40	19 999	37	28	10	310
hepatitis	20	155	20	54	9	348
fd_reduced_15	15	100 000	15	75	3	416
amalgam1	87	50	87	70	4	2737
fd_reduced_30	30	100 000	30	224	3	3436
flight_1k	109	1000	53	161	8	26 652
ncvoter_allc	94	100 000	82	448	15	200 907

Table 4.1: The databases used in the evaluation, ordered by the number of minimal UCCs. Columns and Rows denote the respective dimension of the database, n and m refer to the resulting hypergraph of minimal difference sets, k^* is the transversal rank, that is, the size of the largest minimal UCC.

the code and data available.[14] In some experiments, we collect the run times of intermediate steps, for example the calls to the subroutine (Algorithm 2). To avoid interference with the overall run time measurements, we use separate runs for these. Also, we average over multiple runs to reduce the noise of the measurements. See the corresponding sections for details.

Table 4.1 gives an overview of the data. It lists the number of columns and rows in the database, the number of vertices and edges of the resulting hypergraph, the transversal rank/maximum cardinality of a minimal UCC, as well as the number of solutions. The table is sorted by the number of minimal hitting sets/UCCs. All plots below use this order.

After computing the minimal difference sets, we removed all vertices that do not appear in any edge as they are irrelevant for the enumeration. Therefore, the number n of vertices can be smaller than the number of columns in the database. The particularly stark difference for flight_1k stems from a large

14 hpi.de/friedrich/research/enumdat

portions of the columns being empty. The total number of difference sets of a database with $|r|$ rows is $\binom{|r|}{2}$ in the worst case. However, Table 4.1 shows that the number m of minimal difference sets tends to be much *smaller* than r, let alone quadratic. Put it the other way around, only very few pairs of rows actually contribute to the UCCs and the hypergraph perspective thus provides a very compact representation of the discovery problem. As was observed before by other researchers in data profiling [Abe+18; Köh+16; Pap+15], the maximum cardinality k^* of the minimal UCCs is small in practice. In particular, there does not appear to be any relationship between k^* and the input size.

4.5.2 Run Time, Delay, and Memory

Our enumeration method (Algorithm 1) branches on the vertices in a certain global order. Although the order does not matter for our asymptotic bounds, it does affect the shape of the explored decision tree, which in turn impacts the practical run time. Even on the theoretical side, it has been shown that there exist orders that render already finding the (lexicographically) first solution an **NP**-hard search problem [JPY88]. We are therefore interested how much the branching order influences the performance.

To support the enumeration, we heuristically sort the vertices descendingly by the number of distinct values that appear in the corresponding column of the original database. The intuition is that columns with many values have a higher discriminative power over the pairs of rows and thus are more likely to appear in many minimal UCCs. Including an expressive vertex makes many other vertices obsolete, which should lead to early pruning of the tree. Conversely, excluding such vertices (adding them to the set Y in Algorithm 2) makes it likely that only a few hitting sets survive, which also prunes the tree early. Note that reducing the size of the decision tree, and thus the number of subroutine calls, does not automatically reduce the run time. The remaining calls may have a larger average return time. We discuss this in more detail in Section 4.5.3. As a side note, preliminary experiments showed that sorting the vertices by their hypergraph degree instead (that is, the number of minimal difference sets in which they appear) resulted in similar but slightly worse run times.

Besides using our heuristic order, we also evaluate the algorithms on 1000 random branching orders per dataset. The `ncvoter_allc` instance is an exception, the larger enumeration times do not permit that many experiments. We report on this dataset separately. The run times, averaged over 10 measurements for

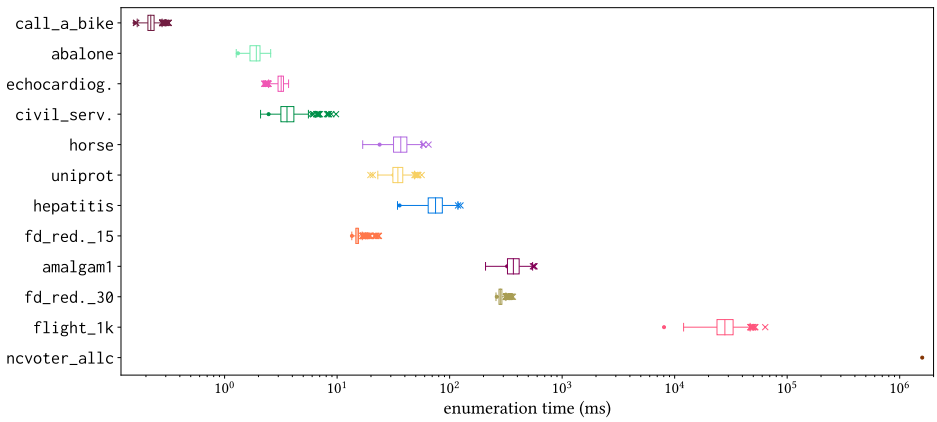


Figure 4.4: Overall run times of the enumeration algorithm. The dot marks the time using the heuristic branching order. For all datasets except `ncvoter_allc`, the box plot shows the times for 1000 random orders. Their median is indicated by a vertical line, the boxes range from the first to third quartile, and the whiskers chart the 1.5 interquartile range above and below those quartiles. Outliers outside of this range are marked by crosses. Each data point is the average over 10 runs.

each data point, are shown in Figure 4.4. The x-axis is scaled logarithmically. The boxes show the first to third quartile of the samples, with the median indicated as a horizontal line. The whiskers represent the smallest data point within 1.5 interquartile range (IQR) of the lower quartile and the highest one within 1.5 IQR of the upper quartile. We count everything beyond that as outliers.

The median run times generally scale with the number of solutions, which is to be expected. They range from 0.25 ms for the 23 minimal UCCs of `call_a_bike` to roughly 27 min for the more than 200k solutions of `ncvoter_allc`. The only exceptions from this trend, that have shorter enumeration times albeit more solutions, are the artificially generated instances `fd_reduced_15` and `fd_reduced_30`. For most of the instances, the branching order had only little impact and the enumeration times are concentrated around the median. Our heuristic outperformed the median random order on all instances, indicating that it is a solid choice in practice. On the `flight_1k` dataset, the heuristic even resulted in a better run time than any of the random orders. For `ncvoter_allc`, however, the influence of the branching order was significantly larger. Using the heuristic, the enumeration completed in less than half an hour. For comparison,

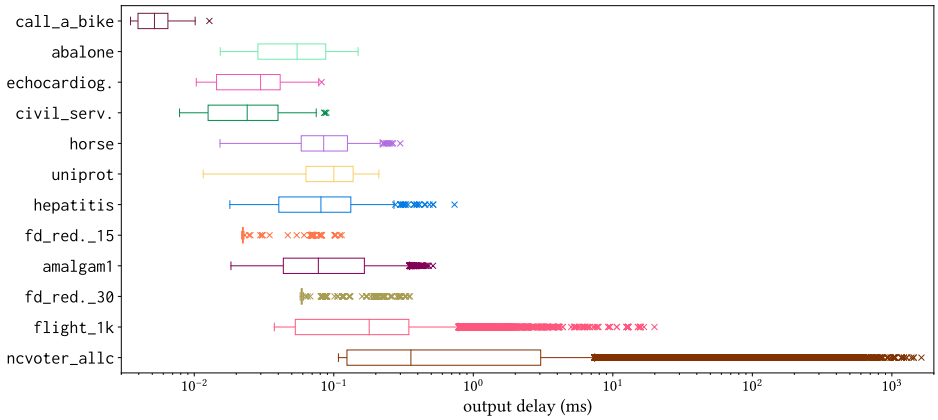
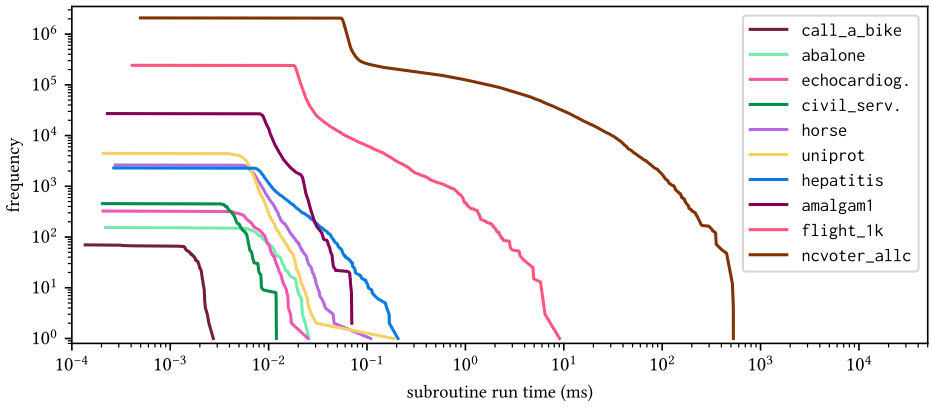


Figure 4.5: Delays between consecutive outputs of minimal unique column combinations using the heuristic branching order. The box plots show the same quartiles as in Figure 4.4. Each data point is the average over 100 runs.

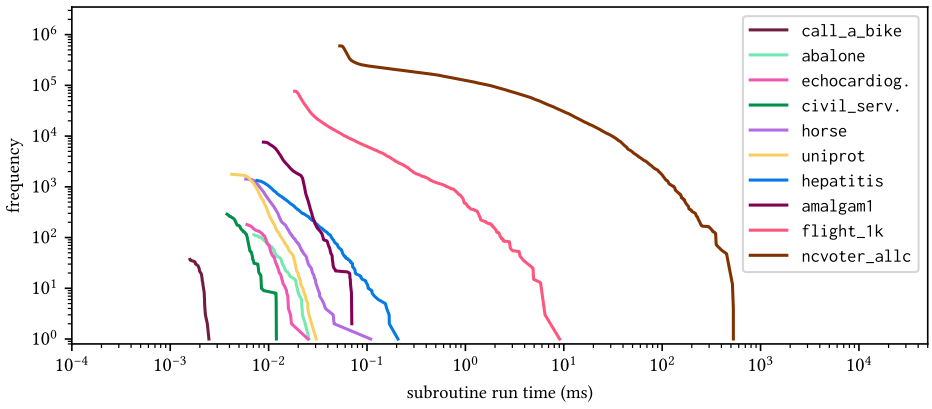
on four out of the eleven random orders we tested, the process only finished after 59.7 h, 105.3 h, 113.7 h, and 167.7 h, respectively. The other seven runs exceeded the time limit of 168 h (one week).

Lemma 4.18 gives a worst-case guarantee on the delay that depends on the maximum size k^* of a minimal UCC. The box plot in Figure 4.5 shows the empirical delays when using the heuristic branching order. Again, the time-axis is logarithmic. Recall that the output order of the solutions is entirely determined by the branching order of the vertices. Each data point in Figure 4.5 corresponds to one output, it was obtained by averaging the delay prior to the same solution over 100 runs. The plot shows that there is a high variance in the delays for the different solutions of an instance. The extreme case is `ncvoter_allc` where the delays range from the order of 10^{-1} ms to over 10^3 ms. Nevertheless, the maximum delay was always less than 2 s, which is reasonably low. The `ncvoter_allc` instance also has the largest solutions with $k^* = 15$. However, the next smaller datasets in that category, `horse` and `uniprot` with transversal ranks of 11 and 10, respectively, have a much lower delay. In general, we cannot confirm a significant correlation between k^* and the empirical delays. In the following section, we investigate the delays more closely by looking at the run time distribution of the calls to the subroutine.

In Lemma 4.18, we also prove a bound on the space requirement, which is



(a)



(b)

Figure 4.6: The complementary cumulative frequencies of the run times of the subroutine calls on the real-world databases using the heuristic branching order. Plot (a) shows all calls, (b) only those entering the brute-force loop in line 11 of Algorithm 2. Each data point is the average of the same call over 100 runs.

independent of the number of solutions. We measured the memory consumption during the enumeration as an average over 5 runs, except for `nc_voter_allc` where we did only a single run. All datasets used between 4.52 MB and 4.68 MB RAM. For comparison, just loading the program without an input takes 4.41 MB. The memory overhead is marginal and independent of the number of solutions. In our experiments, it even seemed to be insensitive to the given instance.

4.5.3 Subroutine Calls

The only potential reasons for super-polynomial delays are the calls to Algorithm 2. It is interesting to examine how many calls we need during the enumeration and how long they actually take in practice. For our heuristic branching order, we measured the run times of each individual call, averaged over 100 runs to reduce the noise. Figure 4.6 shows the *complementary cumulative frequencies* (CCF) of the run times in a log-log plot. That means, for each time t on the x-axis, the plot shows on the y-axis the number of calls with run time at least t. We exclude the artificial instances `fd_reduced_15` and `fd_reduced_30` for now, they are reported separately.

First, we examine the impact of the total number of calls on the run time. The legend of Figure 4.6 is ordered by increasing number of solutions, the same as in the previous plots. For the real-world databases, this is also the same as ordering them by increasing *enumeration time*. For comparison, the total *number* of subroutine calls is marked by the y-value of the left-most endpoint of each curve. The two orders are almost the same. An interesting exception is the `hepatitis` dataset. It has fewer calls than `horse` and `uniprot`, but these calls take more time on average, leading to a higher overall run time. Instance `amalgam1` needs even more calls, which then outweighs the smaller average. Similarly, the calls for `horse` take more time than those for `uniprot`, but the higher number in the latter case causes a longer run time. In preliminary experiments, we observed these effects also when comparing different branching orders for the same dataset. Aiming for a small number of calls is a good strategy, although there are cases where a higher number of easier calls gives a better result.

Next, we discuss the distribution of the calls. The prominent (almost) horizontal lines on the left of Figure 4.6 (a) stem from the few trivial calls with $X = \emptyset$. Those are one to two orders of magnitude faster than all other calls since they do not need to construct the instance of Independent Family. For the non-trivial cases with $X \neq \emptyset$, the extension algorithm first checks whether the

resulting instance can already be decided by the sanity checks in lines 9 and 10 of Algorithm 2. This way, a significant portion of them can be solved in linear time. These calls can be seen in the CCF plots as the steep dip immediately following the horizontal lines. Observe that the y-axis is logarithmic, so the proportion of trivial and easy subroutine calls is significant. Over all databases, slightly more than half of the calls are solved this way. In fact, for the three instances with the most calls, namely, `amalgam1`, `flight_1k`, and `ncvoter_allc`, no more than 32% of the calls entered the brute-force loop in line 11.

This loop is the only part of the algorithm that may requires super-polynomial running time. Figure 4.6 (b) shows the CCFs only for the brute-force calls. The differences between Figures 4.6 (a) and 4.6 (b) in the lower parts of `call_a_bike` and `uniprot` are artifacts of the separate measurements to create these plots. The run times are heterogeneously distributed with many fast invocations and only a few slow ones. As an example, we investigate the calls of the `flight_1k` instance. The database has 1000 rows over 109 columns of which 39 are empty and 17 more do not participate in any *minimal* difference set. The output of `flight_1k` are 26 652 minimal UCCs. During the enumeration process Algorithm 2 is called 242 449 times, 22 (0.009%) calls are trivial, the vast majority of 165 767 (68.4%) are decided easily by the sanity checks, the remaining 76 660 (31.6%) calls enter the loop. Of the brute-force calls, 41 353 (53.9%) take only a *single* iteration to find a suitable combination of candidate private edges verifying that the respective input set X is indeed extendable to a minimal solution (line 13). However, there are also two calls that need the maximum of 74 880 iterations, which corresponds to a run time of 16 ms. In those two cases, all possible combinations of potential witnesses had to be tested, only to conclude that the set is not extendable (line 14). It is inherent to the hardness of Minimal Hitting Set Extension that those inputs that are not extendable because all combinations of candidate private edges cover at least one unhit edge incur the highest number of iterations and thus longest subroutine run times, see Lemma 4.17. Fortunately, those occasions were rare in our experiments. In the case of `flight_1k`, only 622 calls take more than 10 000 iterations, they make up for 0.8% of the brute-force calls and 0.2% of all invocations.

The run time distributions for the other real-world databases are similar to that of `flight_1k`, see Figure 4.6 (a). There is always a non-vanishing chance that any given call to the subroutine incurs a high run time, which is hardly avoidable for a worst-case exponential algorithm, but even the slowest calls are

reasonably fast in practice. However, the majority of calls is far away from the worst case, leading to a very low run time on average. The heterogeneity of the brute-force calls is also showing in the CCFs (Figure 4.6 (b)). They roughly resemble a power-law distribution (straight lines in a log-log plot), albeit their tendency towards small run times (concavity of the plots) is stronger than one would expect for a pure power-law.

Another important point of saving related to the subroutine are those calls that are never actually executed due to the shortcut evaluation in line 7 of Algorithm 1. We compared the implementation as presented here with a version in which this optimization is turned off. Still, the latter version outputs a minimal hitting set as soon as it is found in line 5. We used the heuristic branching order again. Over all real-world instances, the ratio of calls of the non-optimized version that are skipped by the shortcuts is between 12.36% for the `abalone` dataset and 66.11% for `ncvoter_allc`, with a median saving of 37.42%. The skipped calls are those for which we can be certain that the given partial solution is indeed extendable, but not yet minimal. Besides the few calls with $X = \emptyset$, all of them would have entered the brute-force phase to find a suitable set of candidate private edges. On the other hand, they do not need to cycle through all possible combinations and thus are not the hardest calls. A given ratio of skipped calls does not directly translate to a certain time saving. Compared to the enumeration time of the non-optimized version, the shortcuts gain moderate speedup factors from 1.12 for `abalone` up to 2.26 on the `uniprot` dataset, with a median of 1.43.

Finally, the two artificial instances `fd_reduced_15` and `fd_reduced_30` behave very differently from the real-world databases. Figure 4.7 shows their CCFs. The staircase shape indicates that there are only five types of calls, with roughly the same run time for all calls of the same type. Also, the shortcut evaluation hardly saves anything on those datasets. Only 1.90% of the calls for `fd_reduced_15` and 4.97% for `fd_reduced_30` are skipped, resulting in a speedup factor of 1.06 on both instances.

4.6 Conclusion

We devised a backtracking algorithm for the Transversal Hypergraph problem by reducing the enumeration to the **NP**-complete decision whether a set of vertices can be extended to a minimal solution. Although this may seem counterintuitive, it allowed us to reduce both the space usage of the enumeration and

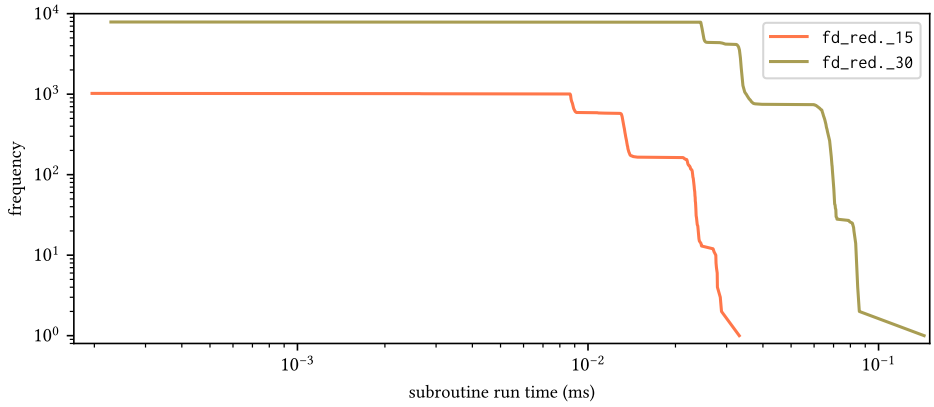


Figure 4.7: The complementary cumulative frequencies of the run times of the subroutine calls on the artificial using the heuristic branching order.

the delay. In particular, we proved that the transversal hypergraph problem can be solved simultaneously with polynomial delay and space on instances whose transversal rank is bounded. We further showed that the extension problem, when parameterized by the size of partial solution, complete for the class **W**[3]. We presented several conditional lower bounds and showed that our extension algorithm is almost optimal assuming SETH. With the nondeterministic generalization of SETH, we identified a complexity-theoretic barrier for closing the remaining gap between our algorithmic results and the lower bounds.

The features of our enumeration method make it particularly suitable for the profiling of relational databases, a domain where the solutions can be expected to be small. Since the size of the largest solution is the degree of the worst-case time bound, it could have been that the run times are still prohibitively large in practice. To guard against such issues, we evaluated our algorithm by discovering the minimal unique column combinations of several real-world and artificially generated databases. The experiments showed that our method succeeds within a reasonable time frame, even when tasked with computing several hundred thousand solutions.

As the empirical run time depends on the branching order of the vertices, we gave a heuristic that achieves good results in practice by reducing the number of calls to the extension subroutine. We also verified that the main reason for the low overall run times is not only the small number of calls but the fact that

the calls are very fast on average. In particular, they regularly avoid the worst case that was the basis for the large theoretical bound.

The tree search underpinning our algorithm obviates the need of expensive coordination between branches or any postprocessing of the solutions. This makes our method easy to implement and memory efficient. Most notably, approaching the discovery of unique column combinations as a hitting set problem resulted in an algorithm that does not need to store previous solutions. This seems to be a major issue even for current state-of-the-art data profiling algorithms such as `DUCC` [Hei+13] and `HyUCC` [PN17]. Papenbrock and Naumann, the authors of `HyUCC`, posed the following challenge [PN17].

> For future work, we suggest to find novel techniques to deal with the often huge amount of results. Currently, `HyUCC` limits its results if these exceed main memory capacity [...].

We will show that this can be solved by viewing data profiling from a hitting set perspective. Until then, there are still some hurdles that need to be overcome to obtain a ready-to-use algorithm. The enumeration phase is the hard core of the problem, but it turns out not to be the true bottleneck in practice. Instead, the quadratic preprocessing step of comparing all pairs of rows in the database regularly took much longer in our experiments than actually enumerating all solutions. Here, careful engineering has the potential of huge speedups on real-world instances. Combining this with the natural advantages of our enumeration approach will yield the novel technique we are looking for.

Our empirical analysis also revealed that the hypergraphs of difference sets are usually much smaller than the input databases. It seems that the hyperedges resulting from most row pairs are non-minimal and therefore do not contribute to the unique column combinations. We investigate these observations more thoroughly in the remainder of the thesis. In Chapter 6, we introduce a model for random hypergraphs that offers an explanation for the small minimizations, but not before we prepare some technical tools in Chapter 5. Finally, we exploit the fast enumeration in Chapter 7 to sidestep the slow preprocessing.

5 A Combinatorial Proof of the Chernoff–Hoeffding Theorem

The Chernoff–Hoeffding theorem estimates the tail of the binomial distribution in terms of the Kullback–Leibler divergence. We give a combinatorial proof for a version in which the upper and lower bounds match up to constant factors. We highlight some applications that fall outside of the topic of hypergraphs.

5.1 Introduction

The minimization, that is, the collection of inclusion-wise minimal edges, of hypergraphs arising from the profiling of relational databases are often much smaller than the databases themselves. In the next chapter, we will conduct a rigorous study of a model for random hypergraphs that shows the same behavior. It will become apparent that the model is closely related to the binomial distribution, the sum of independent Bernoulli trials. For this analysis, we need the *Chernoff–Hoeffding theorem* bounding the tail of that distribution. It states that the distribution function decays exponentially in the distance from its mean. We are mainly interested in the asymptotics of this decay with respect to the number of trials and we need upper and lower bounds that are tight up to constant factors. Such results are usually obtained by approximating the binomial distribution with a normal one and estimating the latter analytically. However, we prefer a combinatorial derivation as this tends to provide some insight on how many trials are needed for the asymptotics to set in. There does not seem to be a version of the theorem readily available in the literature that suits our needs, so we prove it here. Since the Chernoff–Hoeffding theorem has applications also in many other fields of math and science and the proof involves rather lengthy calculations, we present it in a stand-alone manner. We also highlight some consequences that are not related to hypergraphs.

The binomial distribution is ubiquitously used to describe complex systems emerging from the overlapping effects of many independent choices. Its typical behavior is well understood, as described in the central limit theorem and the

strong law of large numbers. Bounding its tails, however, remains the subject of ongoing research. The resulting concentration inequalities play a significant role, for example, in the theory of large deviations [DS89; DZ10], the analysis of randomized algorithms and search heuristics [Doe20; DP09; MU17], and computational learning theory [KV94]. Most of these results can be traced back to the Chernoff–Hoeffding theorem [DP09; Hoe63]. We abbreviate the binary (base-2) logarithm as $\operatorname{ld} x$. Fix a positive integer n and probabilities x and p. Let

$$\mathrm{D}(x \parallel p) = -x \operatorname{ld}\left(\frac{p}{x}\right) - (1-x) \operatorname{ld}\left(\frac{1-p}{1-x}\right)$$

denote the binary *Kullback–Leibler divergence* between the respective Bernoulli distributions, see also Section 2.5. The theorem employs this divergence to bound the probability that a binomially distributed random variable $X \sim \mathrm{Bin}(n, p)$ deviates from its expectation $\mathrm{E}[X] = pn$. Namely, for all $0 \leq x \leq p$, it holds that

$$\mathrm{P}[X \leq xn] \leq 2^{-\mathrm{D}(x \parallel p)n} = \left(\frac{p}{x}\right)^{xn} \left(\frac{1-p}{1-x}\right)^{(1-x)n}.$$

It follows easily that also $\mathrm{P}[X \geq xn] \leq 2^{-\mathrm{D}(x \parallel p)n}$ is true for all $p \leq x \leq 1$ due to the symmetry of X. Several weaker but more practical inequalities have been derived from that, colloquially summarized as Chernoff bounds [Doe20; MU17]. Cramér's theorem asserts that the exponent $\mathrm{D}(x \parallel p)$ is tight [Cra38; DS89]. Any improvement of this inequality can therefore be at most subexponential in n. Stirling's approximation, see [Ash90], gives the following lower bound assuming that the product xn is an integer,

$$\mathrm{P}[X \leq xn] \geq \frac{1}{\sqrt{8nx(1-x)}} \cdot 2^{-\mathrm{D}(x \parallel p)n}.$$

There is an obvious gap of order $\sqrt{n}$ between the two estimates and there seems to be some disparity between the approaches in mathematics and in computer science how to proceed from here. Most of the literature in computer science, for example [Doe20; DP09; MR95; MU17], is satisfied with knowing the order of exponential decay given by $2^{-\mathrm{D}(x \parallel p)n}$ and disregard polynomial factors. In fact, usually bounds that are exponentially weaker than the Chernoff–Hoeffding theorem are used if their application is more convenient. In pure mathematics

on the other hand, not only the polynomial factors are calculated, but the leading coefficients and lower-order terms as well, sometimes even admitting a Taylor-style expansion [Lit69; Mck89; Pro53; Slu77]. All of the latter results are obtained by approximating the binomial distribution via the normal one and analytically estimating the approximation error. We try to strike a balance between the two sides by instead giving a combinatorial proof for bounds that are tight up to constants. In more detail, we show the following.

▶ **Theorem 5.1.** Let n be a positive integer, $0 < p < 1$ a probability, $X \sim \mathrm{Bin}(n, p)$ a binomially distributed random variable. Suppose $x = x(n)$ takes real values in the interval $[\varepsilon,\ 1-\varepsilon]$ for some positive real $\varepsilon > 0$. Let φ and ψ denote

$$\varphi(n,p,x) = \min\left(1,\ \frac{1}{(p-x)\sqrt{xn}}\right) \text{ and } \psi(n,p,x) = \min\left(1,\ \frac{1}{(x-p)\sqrt{(1-x)n}}\right),$$

where we additionally define $\varphi(n,p,p) = 1$ and $\psi(n,p,p) = 1$.

Then, there exist constants $C_1, C_2, C_3, C_4 > 0$, independent of n, x but possibly depending on p, ε, such that the following holds for all n sufficiently large.

(*i*) If $x \le p$, then $C_1\varphi \cdot 2^{-\mathrm{D}(x\,\|\,p)n} \le \mathrm{P}[X \le xn] \le C_2\varphi \cdot 2^{-\mathrm{D}(x\,\|\,p)n}$.

(*ii*) If $x \ge p$, then $C_3\psi \cdot 2^{-\mathrm{D}(x\,\|\,p)n} \le \mathrm{P}[X \ge xn] \le C_4\psi \cdot 2^{-\mathrm{D}(x\,\|\,p)n}$. ◀

Before we commence with the proof, we remark that asymptotics akin to Theorem 5.1 (more precisely, Lemma 5.6) can alternatively be obtained by combining a result of McKay [Mck89, Theorem 2] with an estimate on the complementary cumulative distribution function of the standard normal distribution.

5.2 Integral Case

We split the proof of Theorem 5.1 into two major parts. The first one treats those x for which the product xn is an integer, the second part extends this to the general case. The parts are further subdivided depending on the limiting behavior of $x = x(n)$, namely, whether it converges to p or not. For the integral case, we first use a proposition by Klar [Kla00] about the connection between the binomial distribution function and the probability mass function (PMF).

▶ **Proposition 5.2 (Proposition 1(c) in [Kla00]).** Let n be a positive integers, $p \neq 0$ a probability, and $X \sim \mathrm{Bin}(n,p)$ a binomial variable. For all non-negative integers $k \leq pn$, it holds that

$$1 \leq \frac{\mathrm{P}[X \leq k]}{\mathrm{P}[X = k]} \leq \frac{p(n+1-k)}{n+1-k-(n+1)(1-p)}.$$ ◀

We combine this with the perplexity bound on the binomial coefficient from Proposition 2.7. The explicit form of the lower bounds in the next lemma was known before, see for example the textbook by Ash [Ash90, Lemma 4.7.2], we reprove it here en passant.

▶ **Lemma 5.3.** Let n be a positive integer, $0 < p < 1$ a non-trivial probability, and $X \sim \mathrm{Bin}(n,p)$ a binomial variable. Suppose $0 < x < 1$ is a rational such that xn is an integer.

(*i*) If $x < p$, then

$$\frac{1}{\sqrt{8nx(1-x)}} \cdot 2^{-\mathrm{D}(x\,\|\,p)n} \leq \mathrm{P}[X \leq xn] \leq \frac{p\sqrt{1-x}}{(p-x)\sqrt{\pi xn}} \cdot 2^{-\mathrm{D}(x\,\|\,p)n}.$$

(*ii*) If $p < x$, then

$$\frac{1}{\sqrt{8nx(1-x)}} \cdot 2^{-\mathrm{D}(x\,\|\,p)n} \leq \mathrm{P}[X \geq xn] \leq \frac{(1-p)\sqrt{x}}{(x-p)\sqrt{\pi(1-x)n}} \cdot 2^{-\mathrm{D}(x\,\|\,p)n}.$$ ◀

Proof. Applying the first statement to the variable $\overline{X} \sim \mathrm{Bin}(n, 1-p)$ implies the second statement since $\mathrm{P}[X \geq xn] = \mathrm{P}[\overline{X} \leq (1-x)n]$. Hereby, we use that the Kullback–Leibler divergence observes $\mathrm{D}(1-x\,\|\,1-p) = \mathrm{D}(x\,\|\,p)$.

We are left to prove the first statement. Proposition 2.7 gives the following error bounds on the probability mass function $\mathrm{P}[X = xn] = \binom{n}{xn} \cdot p^{xn}(1-p)^{(1-x)n}$.

$$\frac{1}{\sqrt{8nx(1-x)}} \leq \frac{\mathrm{P}[X = xn]}{2^{\mathrm{H}(x)n} \cdot p^{xn}(1-p)^{(1-x)n}} = \frac{\mathrm{P}[X = xn]}{2^{-\mathrm{D}(x\,\|\,p)\,n}} \leq \frac{1}{\sqrt{\pi nx(1-x)}}.$$

The lower bounds follows immediately from $\mathrm{P}[X \leq xn] \geq \mathrm{P}[X = xn]$.

For the upper bound, we use Proposition 5.2 at the integer position $k = xn$. Let $f_{n,xn}(p)$ denote the resulting bound on the ratio $\mathrm{P}[X \le xn]/\mathrm{P}[X = xn]$,

$$f_{n,xn}(p) = \frac{p\,(n+1-xn)}{n+1-xn-(n+1)(1-p)} = \frac{p\,(n+1-xn)}{p\,(n+1)-xn} = \frac{p\,(1-\frac{xn}{n+1})}{p-\frac{xn}{n+1}}.$$

We claim that for all x and p with $x < p$, the function $f_{n,xn}(p)$ is increasing in n. We show this by verifying that the (partial) discrete derivative $\Delta_n(f_{n,xn})$ with respect to n is positive.

$$\begin{aligned}\Delta_n(f_{n,xn})(p) = f_{n+1,x(n+1)}(p) - f_{n,xn}(p) &= \frac{p\,(n+2-x(n+1))}{p\,(n+2)-x(n+1)} - \frac{p\,(n+1-xn)}{p\,(n+1)-xn} \\ &= \frac{p\,(1-p)\,x}{((p-x)\,n+p)\cdot((p-x)\,n+2p-x)} > 0.\end{aligned}$$

The function $f_{n,xn}(p)$ thus converges from below to $p\,(1-x)/(p-x)$ as n increases, giving an upper bound on $\mathrm{P}[X \le xn]/\mathrm{P}[X = xn]$ for all n. Multiplying with the error bounds and the divergence completes the proof. ■

The upper bounds above are already very close to the desired ones of Theorem 5.1. In fact, we will see that the lemma is enough to conclude $\mathrm{P}[X \le xn] \le \varphi \cdot 2^{-\mathrm{D}(x\,\|\,p)\,n}$ if xn is an integer and $\varphi = \min(1, 1/((p-x)\sqrt{xn}))$. The lower bound in Statement (*i*), however, matches the upper one only if the $x = x(n)$ is bounded away from p for all n. More work is needed for the case $x \to p$. It has already been useful to have a good estimate for the ratio $\mathrm{P}[X \le k]/\mathrm{P}[X = k]$. Unfortunately, Proposition 5.2 gives only a trivial lower bound. We strengthen this in the next lemma, Lemma 5.5 then shows how this translates into a stronger lower bound on the binomial distribution function. Finally, Lemma 5.6 combines all results of this section into a version of the Chernoff–Hoeffding theorem, which is tight whenever the product xn is an integer.

▶ **Lemma 5.4.** Let n be a positive integer, $0 < p < 1$ a non-trivial probability, and $X \sim \mathrm{Bin}(n, p)$ a binomial variable. Then, for all non-negative integers i and k with $i \le k \le pn$, it holds that

$$\frac{\mathrm{P}[X \le k]}{\mathrm{P}[X = k]} \ge (k-i+1)\left(1 - \frac{pn-i}{pn(1-\frac{k}{n})}\right)^{k-i}. \qquad ◀$$

Proof. The PMF of X is increasing for arguments smaller than pn, therefore

$$\frac{\mathrm{P}[X \le k]}{\mathrm{P}[X = k]} = \sum_{j=0}^{k} \frac{\mathrm{P}[X = j]}{\mathrm{P}[X = k]} \ge \sum_{j=i}^{k} \frac{\mathrm{P}[X = j]}{\mathrm{P}[X = k]} \ge (k - i + 1)\, \frac{\mathrm{P}[X = i]}{\mathrm{P}[X = k]}.$$

The last ratio is lower-bounded by

$$\begin{aligned}\frac{\mathrm{P}[X = i]}{\mathrm{P}[X = k]} &= \frac{k!\,(n-k)!}{i!\,(n-i)!}\left(\frac{1-p}{p}\right)^{k-i} \\ &= \prod_{\ell=1}^{k-i} \frac{i+\ell}{n-i-\ell+1} \cdot \left(\frac{1-p}{p}\right)^{k-i} \ge \left(\frac{i}{n-i}\,\frac{1-p}{p}\right)^{k-i}.\end{aligned}$$

For the base of the last expression we get

$$\frac{i}{n-i}\,\frac{1-p}{p} = 1 - \frac{pn-i}{pn-pi} = 1 - \frac{pn-i}{np(1-\frac{i}{n})} \ge 1 - \frac{pn-i}{np(1-\frac{k}{n})}.$$ ■

The first factor $k - i + 1$ increases as i gets smaller while at the same time the second factor decreases. Therefore, in order to apply Lemma 5.4, one has to choose a balancing cut-off point. We do so in the proof of the following lemma.

▶ **Lemma 5.5.** Let n be a positive integer, $0 < p < 1$ a non-trivial probability, and $X \sim \mathrm{Bin}(n, p)$ a binomial variable. Suppose $0 < x < 1$ is a rational such that xn is an integer.

(*i*) If $x < p$, then

$$\mathrm{P}[X \le xn] \ge \frac{p\sqrt{1-x}}{16\sqrt{2}} \cdot \min\left(1,\ \frac{1}{(p-x)\sqrt{xn}}\right) \cdot 2^{-\mathrm{D}(x\,\|\,p)n}.$$

(*ii*) If $p < x$, then

$$\mathrm{P}[X \ge xn] \ge \frac{(1-p)\sqrt{x}}{16\sqrt{2}} \cdot \min\left(1,\ \frac{1}{(x-p)\sqrt{(1-x)n}}\right) \cdot 2^{-\mathrm{D}(x\,\|\,p)n}.$$ ◀

Proof. The second statement follows from the first as in Lemma 5.3. Define an auxiliary integer function $g = g(n, p, x)$ as

$$g(n, p, x) = \left\lfloor \frac{p(1-x)}{2} \cdot \min\left(\sqrt{xn}, \frac{1}{p-x}\right) \right\rfloor.$$

Applying Lemma 5.4 at position $k = xn$ with the cut-off point $i = xn - g$ gives

$$\mathrm{P}[X \le xn] \ge (g+1)\left(1 - \frac{pn - xn + g}{pn(1-x)}\right)^g \cdot \mathrm{P}[X = xn]. \tag{5.1}$$

We want to lower-bound the middle factor in Inequality (5.1) by a constant. Bernoulli's inequality gives

$$\left(1 - \frac{pn - xn + g}{pn(1-x)}\right)^g = \left(1 - \frac{p - x + \frac{g}{n}}{p(1-x)}\right)^g \ge 1 - \frac{g(p-x) + \frac{g^2}{n}}{p(1-x)}.$$

We claim that the numerator $g(p-x) + g^2/n$ is at most $3p(1-x)/4$. We split the argument depending on the relative size of $\sqrt{xn}$ and $1/(p-x)$. If $\sqrt{xn} \ge 1/(p-x)$, then we have $g = \lfloor p(1-x)/2(p-x) \rfloor$ and thus

$$g(p-x) + \frac{g^2}{n} \le \frac{p(1-x)}{2} + \frac{p^2(1-x)^2}{4} \cdot \frac{1}{(p-x)^2 n} \le \frac{p(1-x)}{2} + \frac{p^2(1-x)^2}{4} \cdot x.$$

Conversely, if $\sqrt{xn} \le 1/(p-x)$, then $g = \lfloor p(1-x)\sqrt{xn}/2 \rfloor$ and

$$g(p-x) + \frac{g^2}{n} \le \frac{p(1-x)}{2} \cdot (p-x)\sqrt{xn} + \frac{p^2(1-x)^2}{4} \cdot \frac{xn}{n} \le \frac{p(1-x)}{2} + \frac{p^2(1-x)^2}{4} \cdot x.$$

The last expressions of both inequalities are the same and can be bounded by $3p(1-x)/4$. The middle factor is therefore at least a constant since

$$1 - \frac{g(p-x) + \frac{g^2}{n}}{p(1-x)} \ge 1 - \frac{3p(1-x)}{4} \frac{1}{p(1-x)} = \frac{1}{4}.$$

Reinserting this into Inequality (5.1) and applying the definition of g and Proposition 2.7 gives the result.

$$\begin{aligned}\mathrm{P}[X \le xn] &\ge \frac{g+1}{4} \cdot \mathrm{P}[X = xn] \ge \frac{p(1-x)}{8} \cdot \min\left(\sqrt{xn},\, \frac{1}{p-x}\right) \cdot \mathrm{P}[X = xn] \\ &\ge \frac{p(1-x)}{8} \cdot \min\left(\sqrt{xn},\, \frac{1}{p-x}\right) \cdot \frac{1}{\sqrt{8nx(1-x)}} \cdot 2^{-\mathrm{D}(x\,\|\,p)n} \\ &= \frac{p\sqrt{1-x}}{16\sqrt{2}} \cdot \min\left(1,\, \frac{1}{(p-x)\sqrt{xn}}\right) \cdot 2^{-\mathrm{D}(x\,\|\,p)n}. \qquad \blacksquare\end{aligned}$$

Next, we prove Theorem 5.1 for the case that xn is an integer. We emphasize the facts that Lemma 5.6 holds for all positive integers n, not only asymptotically, and x may range over the whole interval $[0, 1]$.

▶ **Lemma 5.6 (integral case of Theorem 5.1).** Let n be a positive integer, $0 < p < 1$ a non-trivial probability, and $X \sim \mathrm{Bin}(n, p)$ a binomial variable. Suppose $x = x(n)$ takes rational values in the unit interval such that xn is an integer. Let φ and ψ denote the functions

$$\varphi(n,p,x) = \min\left(1,\, \frac{1}{(p-x)\sqrt{xn}}\right) \text{ and } \psi(n,p,x) = \min\left(1,\, \frac{1}{(x-p)\sqrt{(1-x)n}}\right),$$

with additionally $\varphi(n,p,0) = \varphi(n,p,p) = 1$ and $\psi(n,p,1) = \psi(n,p,p) = 1$.

(*i*) If $x \le p$, then $\frac{p\sqrt{1-p}}{16\sqrt{2}} \cdot \varphi \cdot 2^{-\mathrm{D}(x\,\|\,p)n} \le \mathrm{P}[X \le xn] \le \varphi \cdot 2^{-\mathrm{D}(x\,\|\,p)n}$.

(*ii*) If $x \ge p$, then $\frac{(1-p)\sqrt{p}}{16\sqrt{2}} \cdot \psi \cdot 2^{-\mathrm{D}(x\,\|\,p)n} \le \mathrm{P}[X \ge xn] \le \psi \cdot 2^{-\mathrm{D}(x\,\|\,p)n}$. ◀

Proof. Statement (*ii*) follows from (*i*) in the usual way, where we use $\psi(n,p,x) = \varphi(n, 1-p, 1-x)$. Let $C = p\sqrt{1-p}/16\sqrt{2}$. Note that C is at most 0.045 for any p. We first discuss the corner cases $x = 0$ and $x = p$ (assuming that pn is an integer). If $x = 0$, then we have $\mathrm{P}[X \le 0 \cdot n] = (1-p)^n = \varphi(n,p,0) \cdot 2^{-\mathrm{D}(0\,\|\,p)n}$. If $x = p$, the upper bound $\mathrm{P}[X \le pn] \le 1 = \varphi(n,p,p) \cdot 2^{-\mathrm{D}(p\,\|\,p)n}$ holds vacuously. The lower bound follows from pn being the median of the binomial distribution, which implies $\mathrm{P}[X \le pn] \ge 1/2 \ge C = C \cdot \varphi(n,p,p) \cdot 2^{-\mathrm{D}(p\,\|\,p)n}$.

Assume $0 < x < p$. The original Chernoff–Hoeffding theorem and Lemma 5.3 together imply that

$$\begin{aligned} \mathrm{P}[X \le xn] &\le \min\left(1, \frac{p\sqrt{1-x}}{(p-x)\sqrt{\pi xn}}\right) \cdot 2^{-\mathrm{D}(x\,\|\,p)n} \\ &\le \min\left(1, \frac{p}{\sqrt{\pi}}\frac{1}{(p-x)\sqrt{xn}}\right) \cdot 2^{-\mathrm{D}(x\,\|\,p)n} \le \varphi \cdot 2^{-\mathrm{D}(x\,\|\,p)n}. \end{aligned}$$

Finally, the lower bound in this case is an easy consequence of Lemma 5.5 and $p\sqrt{1-x}/16\sqrt{2}$ being larger than $C = p\sqrt{1-p}/16\sqrt{2}$. ∎

5.3 General Case

The second major step of the argument is to extend the result above from integral products xn to arbitrary real x. The equality $\mathrm{P}[X \le xn] = \mathrm{P}[X \le \lfloor xn \rfloor]$ holds universally as X assumes only integer values. In Section 5.2, we have given bounds on the second probability $\mathrm{P}[X \le \lfloor xn \rfloor]$ in terms of the ratio $\lfloor xn \rfloor / n$. To reach the generality of the Chernoff–Hoeffding theorem, we need to infer bounds on $\mathrm{P}[X \le xn]$ in terms of x. In what follows, let x' abbreviate $\lfloor xn \rfloor / n$. Consider the upper bound in Lemma 5.6 (*i*) as an illustrating example. It states

$$\mathrm{P}[X \le xn] = \mathrm{P}[X \le x'n] \le \min\left(1, \frac{1}{(p-x')\sqrt{x'n}}\right) \cdot 2^{-\mathrm{D}(x'\,\|\,p)n}.$$

If there exists some constant C, possibly dependent on p but independent of x and n, such that

$$\frac{1}{(p-x')\sqrt{x'n}} \cdot 2^{-\mathrm{D}(x'\,\|\,p)n} \le \frac{C}{(p-x)\sqrt{xn}} \cdot 2^{-\mathrm{D}(x\,\|\,p)n},$$

then our estimate transfers to the general case; similarly for the other bounds.

The next two lemmas prepare the necessary technical machinery to show the existence of those constants. Lemma 5.7 clarifies the monotonicity of the functions in question. It shows that transitioning from x' to x can only increase the upper bound, meaning that we can actually choose $C = 1$ in the above illustration. For the opposite direction, Lemma 5.8 asserts that this transitions

incurs a multiplicative loss that is at most linear in x. Below, we often conclude the monotonic behavior of a product of functions from that of its factors. While in general the product of non-decreasing functions is not itself non-decreasing, the monotonicity transfers if all factors are additionally non-negative.

▶ **Lemma 5.7.** Let n be a positive integer, and let p and x be two probabilities. The function

$$g_{n,p}(x) = \frac{2^{-\mathrm{D}(x \,\|\, p)n}}{(p-x)\sqrt{xn}}$$

is non-decreasing for all x such that $1/n \le x < p$ for all n is sufficiently large. ◀

Proof. Quantity $2^{-\mathrm{D}(x \,\|\, p)n}$ is non-decreasing for $x \le p$ (Lemma 2.9) and it is not hard to prove this also for $1/(p-x)\sqrt{xn}$ given that $x \ge p/3$. The main focus of this proof is to show that the divergence power dominates the monotonicity of $g_{n,p}$ also for $1/n \le x \le p/3$.

To ease notation, let $\xi = (p-x)\sqrt{xn}$. Taking the derivative of $g_{n,p}$ gives

$$\begin{aligned}
\frac{\mathrm{d}\, g_{n,p}(x)}{\mathrm{d}x} &= \frac{\sqrt{n}}{\xi^2} \cdot \left(\left(\frac{\partial}{\partial x} 2^{-\mathrm{D}(x \,\|\, p)n} \right) (p-x)\sqrt{x} - 2^{-\mathrm{D}(x \,\|\, p)n} \left(\frac{\partial}{\partial x} (p-x)\sqrt{x} \right) \right) \\
&= \frac{\sqrt{n}}{\xi^2} \cdot \left(n \ln\left(\frac{1-x}{x} \frac{p}{1-p} \right) 2^{-\mathrm{D}(x \,\|\, p)n} (p-x)\sqrt{x} - 2^{-\mathrm{D}(x \,\|\, p)n} \frac{p-3x}{2\sqrt{x}} \right) \\
&= \frac{\sqrt{n}\; 2^{-\mathrm{D}(x \,\|\, p)n} (p-x)}{\xi^2\, \sqrt{x}} \cdot \left(n \ln\left(\frac{1-x}{x} \frac{p}{1-p} \right) x - \frac{p-3x}{2(p-x)} \right).
\end{aligned}$$

The first factor is positive for all n and $1/n \le x < p$, the same is true for the second one if $p/3 < x$. Assume $x \le p/3$ in the remainder. Then, the last term of the second factor, $-\frac{p-3x}{2\sqrt{x}}$, is at least $-1/2$. It is thus sufficient to prove the non-negativity of

$$h_{n,p}(x) = n \ln\left(\frac{1-x}{x} \frac{p}{1-p} \right) x - \frac{1}{2}$$

on the subinterval $[1/n,\, p/3]$ for all n large enough. We do this in two claims. First, $h_{n,p}$ is concave there and, secondly, its values at the endpoints of the interval are non-negative. Regarding the concavity, observe that the derivative

$$\frac{\mathrm{d}}{\mathrm{d}x} h_{n,p}(x) = n \ln\left(\frac{1-x}{x} \frac{p}{1-p} \right) - \frac{n}{1-x}$$

is the sum of two non-increasing functions of x; $n \ln(\frac{1-x}{x}\frac{p}{1-p})$ is non-increasing as it is the derivative of the concave mapping $x \mapsto -\mathrm{D}(x \,\|\, p)\, n$. At $1/n$, we have

$$h_{n,p}\left(\frac{1}{n}\right) = \ln\left((n-1)\,\frac{p}{1-p}\right) - \frac{1}{2},$$

which is non-negative for all $n \geq (\sqrt{e}\,(1-p)/p) + 1$. Similarly at endpoint $p/3$,

$$h_{n,p}\left(\frac{p}{3}\right) = n \ln\left(\frac{3-p}{1-p}\right)\frac{p}{3} - \frac{1}{2}$$

is non-negative for $n \geq 3/2p \ln(\frac{3-p}{1-p})$. ■

▶ **Lemma 5.8.** Let n be a positive integer, p and x two probabilities. Define $x' = \lfloor xn \rfloor / n$, and again

$$g_{n,p}(x) = \frac{2^{-\mathrm{D}(x \,\|\, p)n}}{(p-x)\sqrt{xn}}.$$

The following inequalities hold for all x with $1/n \leq x < p$ and n sufficiently large.

(*i*) $g_{n,p}(x') \geq \frac{1-p}{p\sqrt{2}}\, e^{-\frac{1}{2-2p}} \cdot x \cdot g_{n,p}(x)$;

(*ii*) $2^{-\mathrm{D}(x' \,\|\, p)n} \geq \frac{1-p}{p}\, e^{-\frac{1}{2-2p}} \cdot x \cdot 2^{-\mathrm{D}(x \,\|\, p)n}$. ◀

Proof. We make heavy use of the facts that $x - 1/n < x' \leq x$, and that $x \geq 1/n$ implies $x' \geq 1/n$. The relative difference between $g_{n,p}(x')$ and $g_{n,p}(x)$ is

$$\frac{g_{n,p}(x')}{g_{n,p}(x)} = \frac{2^{-\mathrm{D}(x' \,\|\, p)n}}{2^{-\mathrm{D}(x \,\|\, p)n}} \cdot \frac{p-x}{p-x'} \cdot \sqrt{\frac{x}{x'}}.$$

The last factor is at least 1 and the middle one is $\frac{p-x}{p-x'} = 1 - \frac{x-x'}{p-x'} \geq 1 - \frac{1}{pn} \geq \frac{1}{\sqrt{2}}$ given that $n \geq 2/(2-\sqrt{2})p$. Using Lemma 2.9 and $x' - x \leq 1/n$, we get

$$\begin{aligned}
\frac{2^{-\mathrm{D}(x' \,\|\, p)n}}{2^{-\mathrm{D}(x \,\|\, p)n}} &= \left(\frac{x}{1-x}\,\frac{1-p}{p}\right)^{(x-x')n} 2^{-\mathrm{D}(x' \,\|\, x)n} \cdot 2^{-\mathrm{D}(x \,\|\, p)n} \\
&\geq \frac{x}{1-x}\,\frac{1-p}{p}\, 2^{-\mathrm{D}(x' \,\|\, x)n} \cdot 2^{-\mathrm{D}(x \,\|\, p)n} \\
&\geq x \cdot \frac{1-p}{p}\, 2^{-\mathrm{D}(x' \,\|\, x)n} \cdot 2^{-\mathrm{D}(x \,\|\, p)n}.
\end{aligned}$$

It remains to show that $2^{-\mathrm{D}(x' \,\|\, x)n}$ is at least a constant, namely, we claim $2^{-\mathrm{D}(x' \,\|\, x)n} > e^{-\frac{1}{2-2p}}$. Let $t^- = \arg\min_{t \in [x', x]} t(1-t)$, observe that $1/n \le t^- < p$ holds. By Lemma 2.8, the exponent $\mathrm{D}(x' \,\|\, x)\, n$ (to the base $1/2$) is bounded.

$$\begin{aligned} \mathrm{D}(x' \,\|\, x) \cdot n &\le \frac{(x - x')^2}{t^-(1-t^-)\,2\ln 2} \cdot n < \frac{\frac{1}{n^2}}{\frac{1}{n}(1-t^-)\,2\ln 2} \cdot n \\ &= \frac{1}{(1-t^-)\,2\ln 2} < \frac{1}{(2-2p)\ln 2}. \end{aligned}$$

■

We now have the tools ready to prove Theorem 5.1 in its entirety.

▶ **Theorem 5.1 (restated with explicit constants).** Let n be a positive integer, $0 < p < 1$ a non-trivial probability, and $X \sim \mathrm{Bin}(n, p)$ a binomial variable. Suppose the function $x = x(n)$ takes real values in the interval $[\varepsilon, 1-\varepsilon]$ for some $\varepsilon > 0$. Let φ and ψ denote the functions

$$\varphi(n,p,x) = \min\left(1, \frac{1}{(p-x)\sqrt{xn}}\right) \text{ and } \psi(n,p,x) = \min\left(1, \frac{1}{(x-p)\sqrt{(1-x)n}}\right),$$

with additionally $\varphi(n,p,p) = 1$ and $\psi(n,p,p) = 1$. The following statements hold for all n sufficiently large.

(*i*) If $x \le p$, then $\frac{\varepsilon\,(1-p)^{\frac{3}{2}}}{32}\, e^{-\frac{1}{2-2p}} \cdot \varphi \cdot 2^{-\mathrm{D}(x \,\|\, p)n} \le \mathrm{P}[X \le xn] \le \varphi \cdot 2^{-\mathrm{D}(x \,\|\, p)n}$.

(*ii*) If $x \ge p$, then $\frac{\varepsilon\, p^{\frac{3}{2}}}{32}\, e^{-\frac{1}{2p}} \cdot \psi \cdot 2^{-\mathrm{D}(x \,\|\, p)n} \le \mathrm{P}[X \ge xn] \le \psi \cdot 2^{-\mathrm{D}(x \,\|\, p)n}$. ◀

Proof. We only need to prove the first statement. Let $x' = \lfloor xn \rfloor / n$. This implies $x' \le x$ and makes $x'n$ an integer such that $\mathrm{P}[X \le x'n] = \mathrm{P}[X \le xn]$. Recall the definition of $g_{n,p}$ from Lemma 5.7. It is chosen such that for all n, p, and x,

$$\varphi(n,p,x') \cdot 2^{-\mathrm{D}(x' \,\|\, p)n} = \begin{cases} 2^{-\mathrm{D}(x' \,\|\, p)n}, & \text{if } 1 \le \frac{1}{(p-x')\sqrt{x'n}} \text{ or } x' = p; \\ g_{n,p}(x'), & \text{otherwise.} \end{cases}$$

Lemmas 5.7 and 2.9 together establish $\varphi(n,p,x') \cdot 2^{-\mathrm{D}(x' \,\|\, p)n} \le \varphi(n,p,x) \cdot 2^{-\mathrm{D}(x \,\|\, p)n}$ in both cases, provided that n is large enough. The upper bound in Statement (*i*) now follows from the integral case (Lemma 5.6). Regarding the

lower bound, Lemma 5.8 gives for all n large enough,

$$\varphi(n,p,x')\cdot 2^{-\mathrm{D}(x'\,\|\,p)n} \geq \begin{cases} \frac{1-p}{p}\, e^{-\frac{1}{2-2p}} \cdot x \cdot 2^{-\mathrm{D}(x\,\|\,p)n}, & \text{if } 1 \leq \frac{1}{(p-x')\sqrt{x'n}} \text{ or } x' = p; \\ \frac{1-p}{p\sqrt{2}}\, e^{-\frac{1}{2-2p}} \cdot x \cdot g_{n,p}(x), & \text{otherwise.} \end{cases}$$

In summary, using Lemma 5.6 and the assumption $x \geq \varepsilon$, we have

$$\begin{aligned} \mathrm{P}[X \leq xn] = \mathrm{P}[X \leq x'n] &\geq \frac{p\sqrt{1-p}}{16\sqrt{2}} \cdot \varphi(n,p,x') \cdot 2^{-\mathrm{D}(x'\,\|\,p)n} \\ &\geq \frac{\varepsilon\,(1-p)^{\frac{3}{2}}}{32}\, e^{-\frac{1}{2-2p}} \cdot \varphi(n,p,x) \cdot 2^{-\mathrm{D}(x\,\|\,p)n}. \end{aligned}$$ ■

5.4 Applications

In this excursive section, we highlight two applications of Theorem 5.1 that do not fall into the main direction of this thesis, we find them instructive nevertheless. First, we give an alternative proof for the intuition that the probability of a binomial variable taking values outside of a few standard deviations around its expectation may be small but does not converge to 0. The result was previously obtained via the Berry–Esseen inequality using the normal approximation of the binomial distribution [OW15]. Secondly, we improve the rate of convergence in Cramér's theorem for sequences of Bernoulli variables.

Anti-Concentration Inequalities. Chernoff bounds are usually interpreted as concentration inequalities [DP09]. They formalize that the mass of the binomial distribution is concentrated around its mean and that the probability for any other value falls exponentially in the distance to the expectation. However, it is also known that the probability of a binomial random variable $X \sim \mathrm{Bin}(n,p)$ taking values outside of a constant number of standard deviations from $\mathrm{E}[X]$ does not vanish, even as n grows large. Results of the latter kind are occasionally called anti-concentration inequalities. The particular bound[1] we are interested in was given by Oliveto and Witt [OW15]. We reprove it giving a purely combinatorial argument that avoids the normal distribution.

1 Lemma 6.1 in [OW15] only states the existence of a pair of specific constants c, C. It is, however, easy to verify that their proof remains valid for any $c \geq 0$. For a generalization to unequal probabilities, see Lemma 1.10.16 in [Doe20].

▶ **Corollary 5.9 (Lemma 6.1 in [OW15]).** Let n be a positive integer, $0 < p < 1$ a non-trivial probability, and $X \sim \mathrm{Bin}(n,p)$ a binomial variable. Let $\sigma_X = \sqrt{np(1-p)}$ denote the standard deviation of X. For every non-negative real $c \geq 0$, there exists some positive $C > 0$, independent of n but possibly dependent on c and p, such that

$$\mathrm{P}[X \leq \mathrm{E}[X] - c \cdot \sigma_X] \geq C.$$

Conversely, for any non-negative function $f = f(n)$ with $\lim_{n\to\infty} f(n) = \infty$,

$$\lim_{n\to\infty} \mathrm{P}[X \leq \mathrm{E}[X] - f \cdot \sigma_X] = 0. \qquad \blacktriangleleft$$

Proof. Let $c' = c\sqrt{p(1-p)}$. In the notation of the previous sections, we have

$$x = \frac{\mathrm{E}[X] - c \cdot \sigma_X}{n} = p - \frac{c'}{\sqrt{n}}.$$

We assume $x \geq p/2$, this does not loose generality as x converges to p. Applying Theorem 5.1 gives

$$\mathrm{P}[X \leq \mathrm{E}[X] - c \cdot \sigma_X] \geq \frac{p\,(1-p)^{\frac{3}{2}}}{64} e^{-\frac{1}{2-2p}} \cdot \min\left(1, \frac{1}{c'\sqrt{x}}\right) \cdot 2^{-\mathrm{D}(x\,\|\,p)n}.$$

The minimum is at least $1/cp\sqrt{1-p}$. To show that the last factor is bounded, we use an argument similar to the one in Lemma 5.8. Let $t^- = \arg\min_{t\in[x,p]} t(1-t)$. We have $p/2 \leq t^- \leq p$, resulting in

$$\mathrm{D}(x\,\|\,p) \cdot n \leq \frac{\left(\frac{c'}{\sqrt{n}}\right)^2}{t^-(1-t^-)\,2\ln 2} \cdot n = \frac{c^2\,p(1-p)}{t^-(1-t^-)\,2\ln 2} \leq \frac{c^2\,p(1-p)}{\frac{p}{2}(1-p)\,2\ln 2} = \frac{c^2}{\ln 2}.$$

If we move away from $\mathrm{E}[X]$ by more than a constant number of standard deviations, already the original Chernoff–Hoeffding theorem is sharp enough to prove that the distribution function vanishes. For some non-negative $f(n) = \omega(1)$, define $x = p - (f\sqrt{p(1-p)}/\sqrt{n})$. If x is negative, the statement holds vacuously; otherwise, we have $\mathrm{P}[X \leq \mathrm{E}[X] - f \cdot \sigma_X] \leq 2^{-\mathrm{D}(x\,\|\,p)n}$. Let $t^+ =$

$\arg\max_{t\in[x,p]} t(1-t)$. Lemma 2.8 gives

$$\mathrm{D}(x \,\|\, p) \cdot n \geq \frac{f(n)^2\, p(1-p)}{t^+(1-t^+)\,2\ln 2} = \omega(1). \qquad \blacksquare$$

Cramér's Theorem. Theorem 5.1 also has implications for the rate of convergence in Cramér's theorem in the theory of large deviations. Fix some non-trivial probability p and let $(X_i)_i$ be a sequence of independent and identically distributed (i.i.d.) Bernoulli variables[2] with success probability p. Define

$$\Lambda^*(x) = \sup_{t\in\mathbb{R}} \left(tx - \ln\left(\mathrm{E}\left[e^{tX_1}\right]\right)\right)$$

as the Legendre transform of the cumulant-generating function of X_1. Cramér's theorem [Cra38; DS89] states that the transform observes the following convergence for all x with $p < x < 1$,

$$\lim_{n\to\infty} \frac{1}{n} \ln\left(\mathrm{P}\left[\sum_{i=1}^{n} X_i \geq xn\right]\right) = -\Lambda^*(x).$$

Let $\mathrm{D}_e(x \,\|\, y) = \ln(2) \cdot \mathrm{D}(x \,\|\, y) = -x \ln\left(\frac{y}{x}\right) - (1-x)\ln\left(\frac{1-y}{1-x}\right)$ denote the natural (base-e) Kullback–Leibler divergence. It is easy to verify from $\mathrm{E}\left[e^{tX_1}\right] = 1 - p + pe^t$ that Cramér's function $\Lambda^*(x) = \mathrm{D}_e(x \,\|\, p)$ is in fact the natural divergence. This shows that the original Chernoff–Hoeffding inequality with $2^{-\mathrm{D}(x\,\|\,p)n} = e^{-\mathrm{D}_e(x\,\|\,p)n}$ is asymptotically tight up to sublinear terms in the exponent. However, the rate of convergence of the above limit is subject of ongoing research [DS89; DZ10; Fil83]. Theorem 5.1 implies the following corollary.

▶ **Corollary 5.10.** Let $0 < p < 1$ be a non-trivial probability and $(X_i)_i$ a sequence of i.i.d. Bernoulli variables with parameter p. Then, for any x with $p < x < 1$, it holds that

$$\frac{1}{n}\ln\left(\mathrm{P}\left[\sum_{i=1}^{n} X_i \geq xn\right]\right) = -\mathrm{D}_e(x \,\|\, p) - \frac{1}{2}\frac{\ln n}{n} - \frac{\ln(x-p)}{n} - \frac{1}{2}\frac{\ln(1-x)}{n} \pm \mathrm{O}\left(\frac{1}{n}\right).$$

◀

2 Cramér's theorem holds more generally for any i.i.d. sequence $(X_i)_i$ such that the cumulant-generating function of X_1 is finite everywhere [DS89].

6 The Minimization of Random Hypergraphs

We use the Chernoff-Hoeffding theorem to give tight bounds for the expected number of minimal edges of a maximum-entropy random hypergraph. Our results reveal a phase transition in the minimization depending on the number of sampled edges.

6.1 Introduction

We now return to our main topic with an average-case analysis of random hypergraphs. A plethora of work has been dedicated to random graphs, the bibliographies in [Bol01; Hof16] give an overview; hypergraphs, however, received much less attention. For many types of data, they provide a much more natural model. This is especially true if the data has a hierarchical structure or reflects interactions between groups of entities. In non-uniform[1] hypergraphs, where edges can have different numbers of vertices, a phenomenon occurs that is unknown to graphs: an edge may be contained in another, and multiple edges may even form chains of inclusion. We are often only interested in the endpoints of those chains, the collections of inclusion-wise minimal or maximal edges, known as the *minimization* or *maximization* [Ber89], respectively.

We investigate the *maximum-entropy* model for random multi-hypergraphs with n vertices and m edges and expected edge size pn for a constant sampling probability p, which we denote by $\mathcal{B}_{n,m,p}$. In other words, out of all probability distributions on hypergraphs that result in an expected edge size of pn, $\mathcal{B}_{n,m,p}$ is the one of maximum entropy. The notation $\mathcal{B}_{n,m,p}$ is mnemonic of the binomial distribution emerging in the sampling. We are interested in the expected size of the minimization/maximization this hypergraph. Our results are phrased in terms of the minimization, but replacing the probability p with $1-p$ immediately transfers them to the maximization. We show that the size of the minimization undergoes a phase transition with respect to the number of edges m with the

1 A *non-uniform* hypergraph is one whose edges may have different cardinalities, see Section 2.2. This should not be confused with a hypergraph sampled from a non-uniform distribution.

point of transition at $m^* = 1/(1-p)^{(1-p)n}$. While the number of edges is still small, a constant fraction of them is minimal and the minimization grows linearly in the total sample sizes. For $m > m^*$, the size of the minimization is instead governed by the entropy function of the exponent α such that $m = 1/(1-p)^{\alpha n}$ (see Theorem 6.2 for a precise statement). We show that the ratio of minimal edges decreases exponentially as m increases. This seems counter-intuitive at first as it decouples the behavior of the minimization from the growth of the underlying hypergraph. We show that the maximum expected number of minimal edges over all m is of order $\Theta((1+p)^n/\sqrt{n})$, attained at $m = 1/(1-p)^{\frac{n}{1+p}}$.

Our results establish a close connection between the size of the minimization and the binomial distribution. Theorem 6.1 (see below) characterizes the number of minimal edges in terms of the total number of edges m and the likelihood that a binomial variable deviates from its expectation. The main tool in our analysis is the Chernoff–Hoeffding theorem bounding the tail of the distribution function via the Kullback–Leibler divergence from information theory. In the previous chapter, we gave bounds that are tight up to a constant factor, this in turn enables a tight analysis of the minimization.

Our structural insights also have algorithmic implications for the task of actually computing the minimization $\min(\mathcal{H})$ of an input hypergraph $\mathcal{H}$. We give two examples from fine-grained complexity as well as data profiling. There is reason to believe that there exists no minimization algorithm running in time $O(m^{2-\varepsilon}) \cdot \mathsf{poly}(n)$ for any $\varepsilon > 0$ on m-edge, n-vertex hypergraphs. The argument uses the Sperner Family problem, which is to decide whether $\mathcal{H}$ contains two edges such that one is contained in the other, that is, whether $|\min(\mathcal{H})| < |\mathcal{H}|$. Sperner Family is equivalent to the, arguably more prominent, Orthogonal Vectors problem [BCH16] (see also Section 2.4.3). A truly subquadratic algorithm for the minimization would thus falsify the Orthogonal Vectors conjecture (Hypothesis 2.5) and in turn the Strong Exponential Time Hypothesis (Hypothesis 2.3), which would mark a major breakthrough.

On the other hand, partitioning the edges of the hypergraph $\mathcal{H}$ by their cardinality and processing them in increasing order solves the problem in time $O(mn|\min(\mathcal{H})|)$, that is, $O(m^2 n)$ in the worst case. When looking at the average-case complexity for the $\mathcal{B}_{n,m,p}$ distribution, we get $O(mn\,\mathrm{E}[|\min(\mathcal{B}_{n,m,p})|])$. Our results therefore show that the algorithm is subquadratic on average for all m beyond the phase transition; it is even linear for m larger than $1/(1-p)^n$.

There is also a connection to the profiling of relational databases. In the

previous chapters, we have discussed extensively the duality between unique column combinations/functional dependencies and difference sets, ones are the hitting sets of the others. Computing the difference sets one by one generates an incoming stream of seemingly random subsets. Filtering the inclusion-wise minimal ones from the stream does not affect the solution, but can greatly reduce the number of sets and the complexity of the resulting hitting set instance. Minimizing the input is therefore a standard preprocessing technique in data profiling. In real-world databases, there are often fewer minimal difference sets than rows in the database, let alone pairs thereof. Therefore, the space needed to store the sets usually makes up only a small fraction of the original input size. The upper bounds given in Theorem 6.2 provide a theoretical explanation for this observation. We show that only a few difference sets can be expected to be minimal and their number even shrinks as the database grows larger. Conversely, the difference sets and the corresponding multi-column dependencies are mutually dual, which allows to recover the minimized input from the the collection of all solutions. In this sense, the matching lower bounds in Theorem 6.2 can be seen as the smallest amount of data any enumeration algorithm needs to process in order to correctly discover all dependencies.

Related Work. The analysis of random graphs has a long tradition, Erdős–Rényi graphs $\mathcal{G}_{n,m}$ [ER59] and Gilbert graphs $\mathcal{G}_{n,p}$ [Gil59] are arguably the most discussed random graph models in the literature. We refer the reader to the monograph by Bollobás [Bol01] for an overview. A majority of the work on these models concentrates on various phase transitions with respect to the number of edges m or the sample probability p, respectively. This intensive treatment is fueled by the appealing property that Erdős–Rényi graphs are "maximally random" in that they do not assume anything but the number of vertices and edges. Among all probability distributions on graphs with n vertices and m edges, $\mathcal{G}_{n,m}$ is the unique distribution of maximum entropy. The same holds for $\mathcal{G}_{n,p}$ under the constraint that the expected number of edges is $p\binom{n}{2}$, see [AB09a].

The intuition of being maximally random is captured by the Shannon entropy, which is the central concept in information theory [CT06; Sha48]. A discrete stochastic system described by the probability distribution $\vec{p} = (p_i)_{i\in\mathbb{N}}$ has a (binary) entropy of $H((p_i)_{i\in\mathbb{N}}) = -\sum_{i\in\mathbb{N}} p_i \operatorname{ld} p_i$. The self-information of a single state with probability p is $-\operatorname{ld} p$. The entropy is then the expected information of the whole system. It is a measure of surprisal or how "spread out" the distribution is. Originally stemming from thermodynamics [LY99], the

versatility of this definition is key to the successful application of information theory to fields as diverse as cryptography [BF04], machine learning [GB06], quantum computing [NC10], and of course network analysis [New10]. Jaynes' *principle of maximum entropy* states that out of an ensemble of probability distributions that all describe the observed phenomena equally well, the one of maximum entropy is to be preferred in order to minimize any outside bias [Jay57a; Jay57b; Kes09]. Properties of the maximum-entropy model then pertain to the average system matching the observations. In the context of random graphs, it is used to define so-called null models [Zwe14]. After certain graph statistics observed in real-world networks are fixed, one chooses the maximum-entropy distribution that meets these constraints. By comparing the original network with a "typical" graph drawn from the null model, one can infer whether other properties are correlated with the constraints. This method was made rigorous by Park and Newman [PN04] building on earlier work in general statistics. Prescribing the exact or expected number of edges leads to the $\mathcal{G}_{n,m}$ or $\mathcal{G}_{n,p}$ distributions, respectively. The configuration model fixes the whole degree sequence of the graph [Bol80] and the soft configuration model relaxes these constraints to hold in expectation [Bia07; GL08].

Many early attempts to transfer the concept of null models to hypergraphs have been only indirect in that they have studied hypergraphs via their clique expansion [New01] or as bipartite graphs [Sar+15]. This is unsatisfactory since the projections alter relevant observables, like node degrees or the number of triangles. Only recently, Chodrow generalized the configuration model directly to multi-hypergraphs [Cho20], which has subsequently been refined by Arafat et al. [Ara+20]. There also seems to be not much work on hypergraph models that can be cast into the maximum-entropy framework without being intentionally designed as such. A notable exception is the work by Schmidt-Pruzan and Shamir [SS85]. They fix the exact (respectively, expected) edge sequence such that the largest edge has cardinality $O(\log n)$ and show a "double jump" phase transition in the size of the largest connected component. Most of the recent literature on random hypergraphs concentrates on the k-uniform model where every edge has exactly k vertices [BCK10; BCK14; KŁ02] or, equivalently, on random binary matrices with k 1s per column [CFP19]. Our model neither prescribes the exact cardinalities of the edges nor a bound on their maximum size, instead it only requires that the *expected* edge size is pn.

Closest to our work is a string of articles by Demetrovics et al. [Dem+98] as well as Katona [Kat12; Kat13]. They investigate random databases and connect the Rényi entropy[2] of *order 2* of the logarithmic number of rows with the probability that certain unique column combinations or functional dependencies hold. In contrast, we connect the Shannon entropy of the logarithmic number of *pairs* of rows (the quantity α, see Section 6.2) with the expected number of minimal difference sets. Note that the Shannon entropy is the Rényi entropy of *order 1*. In this sense, we complement the result by Demetrovics et al. and show that the duality between UCCs/FDs and difference sets also pertains to the order of entropy. Furthermore, we have observed that in practice the collection of minimal difference sets of real-world databases is much smaller than the original instance. Our findings provide a theoretical explanation for this phenomenon.

The minimization of hypergraphs also occurs in fine-grained complexity in form of the Sperner Family problem. It is subquadratically equivalent to the Orthogonal Vectors problem [BCH16; Gao+18], which in turn admits a fine-grained reduction from CNF-Satisfiability [Wil05]. Any truly subquadratic algorithm for computing the minimization of a hypergraph *in the worst case* would mark a major breakthrough in satisfiability. Very recently, ideas from fine-grained complexity have been extended to to the average case [Bal+17; DLV20; KW18]. We show that a simple algorithm for Sperner Family is subquadratic *on average* on hypergraphs with expected edge size pn.

6.2 Model and Main Results

Fix a probability p and positive integers n and m. The random multi-hypergraph $\mathcal{B}_{n,m,p}$ is generated by independently sampling m (not necessarily distinct) subsets of $[n]$. Each set contains any vertex $v \in [n]$ with probability p independently of all other choices.[3] We quickly argue that it is indeed the maximum-entropy model. Besides the size of the universe n and the number of edges m, the only constraint is the expected edge size pn. The *independence bound* on the entropy reads as follows: Let X_1 to X_m be random variables with joint distribution $\mathrm{P}_{X_1,\dots,X_m}$ and marginals P_{X_j}. Their entropies satisfy $H(\mathrm{P}_{X_1,\dots,X_m}) \leq \sum_{j=1}^{m} H(\mathrm{P}_{X_j})$, with equality holding if and only if the X_j are independent [CT06].

2 Let $\vec{p}$ be a probability distribution and $\alpha \geq 0$, $\alpha \neq 1$, a real number. The *binary Rényi entropy of $\vec{p}$ of order α* is $\frac{1}{1-\alpha} \operatorname{ld}\left(\sum_{i=1}^{n} p_i^{\alpha}\right)$. For $\alpha \to 1$, it converges to the Shannon entropy, see [CT06].

3 In the context of Boolean functions, this is called the *p-biased distribution* [ODo14].

This implies that the edges need to be sampled independently in order to maximize the entropy, and the same is true for the vertices inside one edge. Finally, the fact that setting the sampling probability of the vertices to be all equal indeed yields the maximum entropy under a given mean set size was proven by Harremoës [Har01].

We are interested in the expected number of inclusion-wise minimal edges in $\mathcal{B}_{n,m,p}$, denoted by $\mathrm{E}\big[|\min(\mathcal{B}_{n,m,p})|\big]$. We describe the asymptotic behavior of this expectation with respect to n. That is, we view the number of edges $m = m(n)$ as a function of n and bound the univariate asymptotics of $\mathrm{E}\big[|\min(\mathcal{B}_{n,m,p})|\big]$ in n for any choice of m. The probability p is considered to be a constant throughout.

We show first that the expected size of the minimization can be described precisely in terms of m and the binomial distribution $\mathrm{Bin}(n,p)$. To state our result in full detail, we define

$$\alpha = \log_{\frac{1}{(1-p)^n}} m = -\frac{\log_{1-p} m}{n}.$$

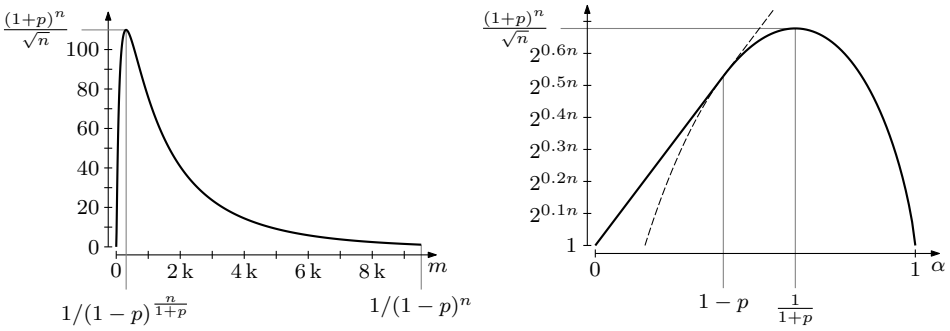
The quantity α is a non-negative function of p, n, and m, it is well-defined for all $0 < p < 1$ and $n,\ m \geq 1$. Asymptotically in n, it is of order $\Theta((\log m)/n)$. If p and n are fixed, choosing a value for α determines m as we have $m = 1/(1-p)^{\alpha n}$.

▶ **Theorem 6.1.** Let n, m be positive integers and p a probability. If $p = 0$ or $p = 1$, then $|\min(\mathcal{B}_{n,m,p})| = 1$ holds deterministically. For $0 < p < 1$, let $X \sim \mathrm{Bin}(n,p)$ be a binomially distributed random variable.

(*i*) For any constant $\varepsilon > 0$ and all $m \leq 1/(1-p)^{(1-\varepsilon)n}$, i.e., all $0 \leq \alpha \leq 1-\varepsilon$, we have $\mathrm{E}\big[|\min(\mathcal{B}_{n,m,p})|\big] = \Theta(m) \cdot \mathrm{P}[X \leq (1-\alpha)n]$.

(*ii*) There exists a constant $c > 0$ such that for all $m \geq 1/(1-p)^{n+c\ln n}$, i.e., all $\alpha \geq 1 + \frac{c \ln n}{n}$, we have $1 \leq \mathrm{E}\big[|\min(\mathcal{B}_{n,m,p})|\big] = 1 + \mathrm{o}(1)$. ◀

The asymptotic estimate in the first statement is tight up to constants, the second statement is even tight up to lower-order terms. The constants hidden in the big-O notation are universal in the sense that they do not depend on m or n, but also not on α, which describes the relation between the former two. We note that the constants may depend on p and ε. There is a gap in the theorem at $m = 1/(1-p)^n$, which can, however, be made arbitrarily small. Let $C = 1/(1-p)$, then Statement (*i*) holds if $m \leq (C-\delta)^n$ for any constant $\delta > 0$ and Statement (*ii*) takes over at $m \geq (C+\mathrm{o}(1))^n$. Unlike what one might expect, the characterization



(a) The expected size of the minimization as a function of m for $n = 10$ and $p = 0.6$ in the information-theoretic regime. The vertical line at $m = 1/(1-p)^{\frac{n}{1+p}}$ indicates the maximum (Lemma 6.12). For $m > 1/(1-p)^n$, the size goes to 1. The linear bound for $m \le 1/(1-p)^{(1-p)n}$ is not shown as it is too close to 0.

(b) The expected size of the minimization as a function of α for $p = 0.6$. The plot is independent of n. The vertical line at $\alpha = 1-p$ indicates the phase transition between the linear and the information-theoretic regime. The respective bounds are continued as dashed lines into the other regime. The vertical line at $\alpha = 1/(1+p)$ indicates the maximum (Lemma 6.12).

Figure 6.1: Illustration of Theorem 6.2 showing the expected size of the minimization of a random hypergraph depending (a) on the number of edges m and (b) on α. As α is logarithmic in m, (b) is the same plot as (a), but with both axes being logarithmic.

in terms of the binomial distribution in Theorem 6.1 (*i*) cannot be extended to the case in which α converges to 1. We prove this in Lemma 6.11.

Theorem 6.1 allows us to express the size of the minimization in information-theoretic terms, namely, in terms of the Shannon entropy of α. This reveals a phase transition at $m^* = 1/(1-p)^{(1-p)n}$, i.e., for $\alpha = 1-p$. Recall from Section 2.5 that $\mathrm{H}(x) = -x \operatorname{ld} x - (1-x)\operatorname{ld}(1-x)$ is the (binary Shannon) entropy function.

▶ **Theorem 6.2.** Let n, m be positive integers and $0 < p < 1$ a probability.

(*i*) If $m \le 1/(1-p)^{(1-p)n}$, that is $\alpha \le 1-p$, then $\mathrm{E}\big[|\min(\mathcal{B}_{n,m,p})|\big] = \Theta(m)$.

(*ii*) Let $\varepsilon > 0$ be any constant. If m is between $1/(1-p)^{(1-p)n}$ and $1/(1-p)^{(1-\varepsilon)n}$, that is, $1-p \le \alpha \le 1-\varepsilon$, then $\mathrm{E}\big[|\min(\mathcal{B}_{n,m,p})|\big]$ is of order

$$\Theta(1) \cdot \min\left(1, \frac{1}{(\alpha-(1-p))\sqrt{(1-\alpha)n}}\right) \cdot 2^{(\mathrm{H}(\alpha)+(1-\alpha)\log_2 p)\,n}$$

$$= \Theta(1) \cdot \min\left(1, \frac{1}{(\alpha - (1-p))\sqrt{(1-\alpha)n}}\right) \cdot \left(\frac{p^{1-\alpha}}{(1-\alpha)^{1-\alpha}\,\alpha^{\alpha}}\right)^{n}. \quad \blacktriangleleft$$

The leading constant in Statement (*ii*) may depend on ε. We derive Theorem 6.2 from Theorem 6.1 by applying the tightened variant of the Chernoff–Hoeffding theorem, Theorem 5.1, which we proved in the previous chapter.

The bounds in the two cases are very different. They are visualized in Figure 6.1 showing the expected size of the minimization as a function of the number of edges m and of the exponent α, respectively. To distinguish the cases also in writing, we use the term *linear regime* if m is between 1 and $1/(1-p)^{(1-p)n}$, corresponding to $0 \leq \alpha \leq 1-p$. The number of minimal edges is proportional to the total number of edges in that regime. Likewise, we refer to m being between $1/(1-p)^{(1-p)n}$ and $1/(1-p)^{n}$, i.e., $1-p \leq \alpha \leq 1$, as the *information-theoretic regime*. There, the size of minimization can be described using the entropy of α. It first increases to its maximum and afterwards decays rapidly. In particular, larger hypergraphs have *fewer* minimal edges. This is in line with our observations regarding hypergraphs from real-world databases, see Chapter 4.

6.3 Distinct Sets and Minimality

We want to determine the expected number of minimal edges in the maximum-entropy multi-hypergraph $\mathcal{B}_{n,m,p}$. The vertex sampling probabilities $p = 0$ or $p = 1$ result in trivial hypergraphs containing m copies of the edge $\emptyset$ or $[n]$, respectively. The minimization then consist of a single edge. In the remainder of this chapter, we thus assume $0 < p < 1$. Also, n and m always denote positive integers and $X \sim \mathrm{Bin}(n, p)$ a binomial variable. As the sampling probability p is non-trivial, every subset of $[n]$ then has a non-vanishing chance to be sampled. A set is minimal for $\mathcal{B}_{n,m,p}$ if and only if it is generated in one of the trials and no proper subset ever occurs. Both aspects influence the chance of minimality, but their impact varies depending on the cardinality of the sampled sets.

The number of vertices per edge is heavily concentrated around the mean pn and the more vertices there are in an edge, the less likely it is minimal. Intuitively, almost no sets with very low cardinalities are sampled, but if so, they are very likely included in the minimization. There are plenty of edges with a medium number of vertices and there is still a good chance they are minimal. Sets of high cardinality rarely occur and usually they are dominated by smaller edges. This

disparity is exacerbated by a large number of trials. Boosting m increases the probability that also sets of cardinality a bit further away from pn are sampled, at the same time the process now generates more duplicate sets that do not count towards the minimization. More importantly though, the likelihood of a larger set being minimal is much smaller with many trials. Eventually, the last effect outweighs all others, in the extreme case the only minimal edge is empty.

We make this intuition rigorous in this section. We give preliminary bounds on the number of minimal edges as a first step towards Theorem 6.1. The estimates are already tight up to constants but are rather unwieldy. They will serve as the basis for our further analysis. In order to prove the bounds, we prepare the following technical lemma on independent repetitions of random experiments.

▶ **Lemma 6.3.** Consider a random experiment with outcomes A, B, and C, where $\mathrm{P}[B] > 0$. In m independent and identically distributed trials, let A_j be the event that the outcome of the j-th trial is A, same with B. It holds that

$$\mathrm{P}\big[\forall j \le m\colon \neg A_j \,\big|\, \exists k \le m\colon B_k\big] \le \mathrm{P}\big[\forall j \le m\colon \neg A_j \,\big|\, B_m\big] = \mathrm{P}\big[\forall j < m\colon \neg A_j\big] \qquad \blacktriangleleft$$

Proof. The equality is immediate from the independence of the trials and B_m implying $\neg A_m$. We only have to prove the inequality. There, the case $\mathrm{P}[B] = 1$ is trivial, thus assume $0 < \mathrm{P}[B] < 1$ in the following. We first show that the inequality is equivalent to

$$\mathrm{P}\big[\forall j \le m\colon \neg A_j \,\big|\, \neg B_m \wedge (\exists k < m\colon B_k)\big] \le \mathrm{P}\big[\forall j \le m\colon \neg A_j \,\big|\, B_m\big]. \tag{6.1}$$

Event $[\exists k \le m\colon B_k]$ can be partitioned into $[\neg B_m \wedge (\exists k < m\colon B_k)]$ and $[B_m]$.

$$\begin{aligned}\mathrm{P}\big[\forall j \le m\colon \neg A_j \,\big|\, \exists k \le m\colon B_k\big] &= \frac{\mathrm{P}\big[(\forall j \le m\colon \neg A_j) \wedge (\exists k \le m\colon B_k)\big]}{\mathrm{P}[\exists k \le m\colon B_k]} \\ &= \frac{\mathrm{P}\big[(\forall j \le m\colon \neg A_j) \wedge \neg B_m \wedge (\exists k < m\colon B_k)\big] + \mathrm{P}\big[(\forall j \le m\colon \neg A_j) \wedge B_m\big]}{\mathrm{P}[\neg B_m \wedge (\exists k < m\colon B_k)] + \mathrm{P}[B_m]}.\end{aligned}$$

Observe that for any four reals x, y, z, w such that y, w, and $y+w$ are non-zero, $\frac{x+z}{y+w} \le \frac{z}{w}$ holds if and only if $\frac{x}{y} \le \frac{z}{w}$ does. Applying this to the real numbers $x = \mathrm{P}\big[(\forall j \le m\colon \neg A_j) \wedge \neg B_m \wedge (\exists k < m\colon B_k)\big]$, $y = \mathrm{P}[\neg B_m \wedge (\exists k < m\colon B_k)]$, $z = \mathrm{P}\big[(\forall j \le m\colon \neg A_j) \wedge B_m\big]$, and $w = \mathrm{P}[B_m]$ gives the equivalence.

The actual lemma is proven by induction over m. The case $m = 1$ is trivial. Suppose that $\mathrm{P}\left[\forall j < m\colon \neg A_j \,\middle|\, \exists k < m\colon B_k\right] \leq \mathrm{P}\left[\forall j < m\colon \neg A_j \,\middle|\, B_{m-1}\right]$ holds. It is now sufficient to conclude Inequality (6.1). The independence of the trials implies that the probability $\mathrm{P}\left[\forall j \leq m\colon \neg A_j \,\middle|\, \neg B_m \wedge (\exists k < m\colon B_k)\right]$ is equal to

$$\begin{aligned}
&\frac{\mathrm{P}\left[(\forall j \leq m\colon \neg A_j) \wedge \neg B_m \wedge (\exists k < m\colon B_k)\right]}{\mathrm{P}[\neg B_m \wedge (\exists k < m\colon B_k)]} \\
&\quad = \frac{\mathrm{P}[\neg A_m \wedge \neg B_m] \cdot \mathrm{P}\left[(\forall j < m\colon \neg A_j) \wedge (\exists k < m\colon B_k)\right]}{\mathrm{P}[\neg B_m] \cdot \mathrm{P}[\exists k < m\colon B_k]} \\
&\quad = \mathrm{P}[\neg A_m \mid \neg B_m] \cdot \mathrm{P}\left[\forall j < m\colon \neg A_j \,\middle|\, \exists k < m\colon B_k\right].
\end{aligned}$$

By induction, the latter is at most $\mathrm{P}[\neg A_m \mid \neg B_m] \cdot \mathrm{P}\left[\forall j < m\colon \neg A_j \,\middle|\, B_{m-1}\right]$. The probabilities of the outcomes do not change over the trials, and also event B_m implies $\neg A_m$. Therefore, $\mathrm{P}[\neg A_m \mid \neg B_m] \cdot \mathrm{P}\left[\forall j < m\colon \neg A_j \,\middle|\, B_{m-1}\right]$ equals

$$\begin{aligned}
&\frac{1 - \mathrm{P}[A_m] - \mathrm{P}[B_m]}{1 - \mathrm{P}[B_m]} \, \frac{(1 - \mathrm{P}[A_m])^{m-2} \cdot \mathrm{P}[B_{m-1}]}{\mathrm{P}[B_{m-1}]} \\
&\quad = \left(1 - \frac{\mathrm{P}[A_m]}{1 - \mathrm{P}[B_m]}\right) \cdot \left(1 - \mathrm{P}[A_m]\right)^{m-2} \leq \left(1 - \mathrm{P}[A_m]\right)^{m-1} \\
&\quad = \mathrm{P}\left[\forall j \leq m\colon \neg A_j \,\middle|\, B_m\right].
\end{aligned}$$

■

We now prove the preliminary estimates for $\mathrm{E}\left[|\min(\mathcal{B}_{n,m,p})|\right]$. They are binomial sums of polynomials of probabilities. Depending on which factors we include in the sum, we get an upper or lower bound. Let $\mathcal{D}_{n,p}$ be the maximum-entropy distribution on the power set $\mathcal{P}([n])$ with $\mathrm{E}_{S \sim \mathcal{D}_{n,p}}[|S|] = pn$, namely, each vertex is included independently with probability p. Let $S_j \sim \mathcal{D}_{n,p}$ be the j-th sampled edge. Define the function

$$s_{n,p}(i,m) = \mathrm{P}\left[\exists j \leq m\colon S_j = [i]\right] = 1 - (1 - p^i(1-p)^{n-i})^m$$

as the probability that the set $[i]$ (in fact, any set of cardinality i) is generated. For the closed forms of $s_{n,p}(i,m)$, observe that the random set S_j differs from $[i]$ with probability $1 - p^i(1-p)^{n-i}$. In the same fashion, let

$$w_{n,p}(i,m) = \mathrm{P}\left[\forall j \leq m\colon \neg(S_j \subsetneq [i])\right] = (1 - (1-p)^{n-i}(1-p^i))^m$$

be the probability that no trial results in a proper subset of $[i]$.[4] The expression for $w_{n,p}(i,m)$ follows from the fact that S_j is a subset of $[i]$ if it does not contain an element of $[n]\backslash[i]$, which has probability $(1-p)^{n-i}$. Conditioned on being *any* subset, S_j is a *proper* subset if it is missing at least one element of $[i]$.

▶ **Lemma 6.4.** The following statements holds for the minimization $\mathcal{B}_{n,m,p}$.

(*i*) $\mathrm{E}\big[|\min(\mathcal{B}_{n,m,p})|\big] \geq \sum_{i=0}^{n} \binom{n}{i} s_{n,p}(i,m) \cdot w_{n,p}(i,m)$.

(*ii*) $\mathrm{E}\big[|\min(\mathcal{B}_{n,m,p})|\big] \leq \sum_{i=0}^{n} \binom{n}{i} s_{n,p}(i,m) \cdot w_{n,p}(i,m-1)$.

(*iii*) $\mathrm{E}\big[|\min(\mathcal{B}_{n,m,p})|\big] \leq 1 + \frac{1}{p} \sum_{i=0}^{n} \binom{n}{i} s_{n,p}(i,m) \cdot w_{n,p}(i,m)$. ◀

Proof. Some fixed set $S \subseteq [n]$ is in $\min(\mathcal{B}_{n,m,p})$ if and only if it is sampled in one of the m trials and no proper subset is sampled. The probability for both events depends only on $|S|$ as all sets with the same cardinality are equally likely.

$$\mathrm{E}\big[|\min(\mathcal{B}_{n,m,p})|\big] = \sum_{S\subseteq[n]} \mathrm{P}\big[(\exists k \leq m\colon S_k = S) \wedge (\forall j \leq m\colon \neg(S_j \subsetneq S))\big]$$

$$= \sum_{i=0}^{n} \binom{n}{i} \cdot \mathrm{P}[\exists k \leq m\colon S_k = [i]] \cdot \mathrm{P}\big[\forall j \leq m\colon \neg(S_j \subsetneq [i]) \mid \exists k \leq m\colon S_k = [i]\big]$$

$$= \sum_{i=0}^{n} \binom{n}{i} s_{n,p}(i,m) \cdot \mathrm{P}\big[\forall j \leq m\colon \neg(S_j \subsetneq [i]) \mid \exists k \leq m\colon S_k = [i]\big].$$

The last factor is the likelihood that any set with i elements is minimal, if it is sampled at all. The bounds differ only in the way this factor is estimated. We claim it is at least $w_{n,p}(i,m) = \mathrm{P}\big[\forall j \leq m\colon \neg(S_j \subsetneq [i])\big]$ (without the condition) and also at most $w_{n,p}(i,m-1) = \mathrm{P}\big[\forall j < m\colon \neg(S_j \subsetneq [i])\big]$ (with one fewer trial).

The first inequality is obvious because conditioning on some trial producing $[i]$ itself only increases the chances of never sampling a proper subset. For the second inequality, we apply Lemma 6.3 to the events $A_j = [S_j \subsetneq [i]]$ and $B_j = [S_j = [i]]$, showing that $\mathrm{P}\big[\forall j \leq m\colon \neg(S_j \subsetneq [i]) \,\big|\, \exists k \leq m\colon S_k = [i]\big]$ is indeed upper-bounded by $\mathrm{P}\big[\forall j \leq m\colon \neg(S_j \subsetneq [i]) \mid S_m = [i]\big] = \mathrm{P}\big[\forall j < m\colon \neg(S_j \subsetneq [i])\big]$. This completes the proof of Statements (*i*) and (*ii*).

For Statement (*iii*), note that the i-th term of the respective sums in the first two statements have ratio $w_{n,p}(i,m)/w_{n,p}(i,m-1)$. Due to the independence,

4 The notation $s_{n,p}$ refers to the set being *sampled*, these probabilities are then *weighted* by $w_{n,p}$.

this is equal to $w_{n,p}(i,1)$, the probability to not sample a strict subset of $[i]$ in a single trial. For $i < n$, it is easy to see that $w_{n,p}(i,1) \geq p$. If $i = n$, we have $w_{n,p}(n,1) = p^n$ which converges to 0. However, the statement follows anyway as the contribution of the last term to the whole sum is at most 1. ■

The part that all three bounds of Lemma 6.4 have in common describes the expected number of *distinct* sets in $\mathcal{B}_{n,m,p}$. Recall that we use $\|\mathcal{H}\|$ to denote the number of distinct sets of some multi-hypergraph $\mathcal{H}$. That means, we have

$$\mathrm{E}\big[\|\mathcal{B}_{n,m,p}\|\big] = \sum_{i=0}^{n} \binom{n}{i} s_{n,p}(i,m) = \sum_{i=0}^{n} \binom{n}{i} \Big(1 - (1 - p^i(1-p)^{n-i})^m\Big).$$

We weight the terms by $w_{n,p}(i,m)$ or $w_{n,p}(i,m-1)$ to estimate the size of the minimization. We analyze the two parts separately, starting with function $w_{n,p}$.

The behavior of the *weighting factors* $w_{n,p}$ may have applications besides our study of random hypergraphs. Consider m trials according to the maximum-entropy distribution $\mathcal{D}_{n,p}$ on subsets of $[n]$. Then, $w_{n,p}(i,m)$ is by definition the probability that any fixed subset of cardinality i survives as minimal after m trials, equivalently, any proper subset is sampled with probability $1 - w_{n,p}(i,m)$. It is easy to see that weighting factors are non-increasing in both i and m. We prove next that the weighting factors are in fact threshold functions falling abruptly from almost 1 to almost 0 as i increases from 0 to n, the position of the transition depends on n, m, and p. Recall that $\alpha = -(\log_{1-p} m)/n$. Lemma 6.5 below establishes a sharp threshold behavior of $w_{n,p}(i,m)$ at

$$i^* = n + \log_{1-p} m = (1-\alpha)\, n.$$

Note that i^* is always at most n since $\log_{1-p} m$ is non-positive. The definition is such that it ensures the equality $m = 1/(1-p)^{\alpha n} = 1/(1-p)^{n-i^*}$. For increasing m, the threshold gets smaller relative to n. Once m grows beyond $1/(1-p)^n$, i.e., $\alpha > 1$, the quantity i^* can no longer be interpreted as a cardinality as it becomes negative. Later, in Lemma 6.10, we will see that m being this large is in fact irrelevant for the analysis of the minimization.

▶ **Lemma 6.5.** It holds that $w_{n,p}(0,m) = 1$ and $w_{n,p}(n,m) = p^{nm}$.
Now suppose $i = i(n)$ takes integer values strictly between 0 and n.

(*i*) The factor $w_{n,p}(i,m)$ is at least $\exp(-m(1-p)^{n-i}) \cdot (1 - m(1-p)^{2(n-i)})$ and at most $\exp(-m(1-p)^{n-i+1})$.

In particular, the following statements hold.[5]

(*ii*) If $i = n + \log_{1-p} m + \omega(1)$, then $\lim_{n\to\infty} w_{n,p}(i,m) = 0$.

(*iii*) If $i = n + \log_{1-p} m - \omega(1)$, then $\lim_{n\to\infty} w_{n,p}(i,m) = 1$.

(*iv*) If $i = n + \log_{1-p} m \pm \Theta(1)$, then $w_{n,p}(i,m) = \Theta(1)$. ◀

Proof. The corner cases for $i = 0$ and $i = n$ are elementary. Assume $0 < i < n$ for the rest of the proof. We use Corollary 2.11 to derive both the upper and the lower bound for $w_{n,p}(i,m)$. On the one hand, it yields

$$\begin{aligned} w_{n,p}(i,m) &= (1 - (1-p)^{n-i}(1-p^i))^m \le (1 - (1-p)^{n-i}(1-p))^m \\ &\le \exp\Big(-m(1-p)^{n-i} \cdot (1-p)\Big). \end{aligned}$$

On the other hand, we have $1 - p^i \le 1$ and get that

$$w_{n,p}(i,m) \ge (1 - (1-p)^{n-i})^m \ge \exp\Big(-m(1-p)^{n-i}\Big) \cdot (1 - m(1-p)^{2(n-i)}).$$

The above shows that the limiting behavior of $w_{n,p}(i,m)$ as n grows large is entirely determined by the product $m(1-p)^{n-i}$. If $i = n + \log_{1-p} m + \omega(1)$, then

$$m(1-p)^{n-i} = m(1-p)^{-(\log_{1-p} m) - \omega(1)} = (1-p)^{-\omega(1)}$$

diverges and $w_{n,p}(i,m)$ thus converges to 0. If $i = n + \log_{1-p} m - \omega(1)$, both $m(1-p)^{n-i} = (1-p)^{\omega(1)}$ and

$$m(1-p)^{2(n-i)} = m(1-p)^{-2(\log_{1-p} m) + \omega(1)} = \frac{(1-p)^{\omega(1)}}{m}$$

tend to 0, implying $\lim_{n\to\infty} w_{n,p}(i,m) = 1$.

5 We understand $\omega(1)$ as the class of all non-negative unbounded functions of n. In particular, the classes $n + \log_{1-p} m + \omega(1)$ and $n + \log_{1-p} m - \omega(1)$ are disjoint.

Finally, if the cardinality i is close to the threshold $i^* = n + \log_{1-p} m$, the limit may not exist. We show that $w_{n,p}(i,m)$ is still bounded away from 0 for all n. Suppose $i = n + \log_{1-p} m \pm \Theta(1)$. In particular, this means that the difference $i^* - i$ is bounded. If $m = m(n)$ is (at most) a constant, the same holds for $w_{n,p}(i,m) \geq (1-(1-p)^{n-i})^m \geq p^m$. Here, we used the assumption $i < n$. If m diverges instead, then $n - i = \log_{1-p} m \mp \Theta(1) = \omega(1)$ diverges with it and thus

$$\begin{aligned} w_{n,p}(i,m) &\geq \exp\left(-m(1-p)^{n-i}\right)\left(1-m(1-p)^{2(n-i)}\right) \\ &= \exp\left(-(1-p)^{i^*-i}\right) \cdot \left(1-(1-p)^{(i^*-i)+(n-i)}\right) = \Omega(1). \end{aligned}$$

■

We will later need the following stronger form of Statement (*iv*).

▶ **Corollary 6.6.** Let $i = \lfloor n + \log_{1-p} m \rfloor$. There exists a constant $\delta > 0$ such that for all m with $1 < m \leq 1/(1-p)^n$, we have $w_{n,p}(i,m) \geq \delta$. ◀

Proof. The assumption on m is to ensure that $0 \leq i < n$. If $i = 0$, we have $w_{n,p}(i,m) = 1$. Therefore, the minimum of $w_{n,p}(i,m)$ over all n and every choice of function $m = m(n)$ is attained at a position where $i > 0$. It holds that

$$\begin{aligned} w_{n,p}(i,m) &\geq \exp\left(-m(1-p)^{n-i}\right) \cdot \left(1-m(1-p)^{2(n-i)}\right) \\ &= \exp\left(-m(1-p)^{n-\lfloor n+\log_{1-p} m\rfloor}\right) \cdot \left(1-m(1-p)^{2(n-\lfloor n+\log_{1-p} m\rfloor)}\right) \\ &\geq \exp(-1) \cdot \left(1-\frac{1}{m}\right) \geq \frac{1}{2e}. \end{aligned}$$

■

We have demonstrated a sharp threshold for the weighting factors. We now move our attention to the number of distinct sets $\|\mathcal{B}_{n,m,p}\|$. A trivial cap for that is the total number of edges m. When starting the sampling, many different sets are generated and $\|\mathcal{B}_{n,m,p}\|$ is close to m. With more and more trials though, duplicates occur in the sample and the two quantities grow apart. To discuss this in more detail, we introduce some notation. For integers ℓ, u with $0 \leq \ell \leq u \leq n$, let $\|\mathcal{B}_{n,m,p}(\ell,u)\|$ denote the number of distinct samples whose cardinality is between ℓ and u, including. This number is of course at most as large as the total number of samples in that range. It thus makes sense to expect an upper bound closely related to the binomial distribution. We confirm this intuition below and also prove a lower bound for $\|\mathcal{B}_{n,m,p}(\ell,u)\|$ of the same flavor.

Recall, that $X \sim \mathrm{Bin}(n,p)$ denotes a binomial random variable. To state the lemma, we further define $\mathbf{p} = \mathbf{p}(\ell,u) = \max_{\ell \le i \le u}\{p^i(1-p)^{n-i}\}$. It holds that

$$\mathbf{p} = \begin{cases} p^\ell(1-p)^{n-\ell}, & \text{if } p < 1/2; \\ 1/2^n, & \text{if } p = 1/2; \\ p^u(1-p)^{n-u}, & \text{otherwise.} \end{cases}$$

The closed form for $\mathbf{p}$ follows from the equality $p^i(1-p)^{n-i} = (p/(1-p))^i \cdot (1-p)^n$ since the odds $p/(1-p)$ are strictly smaller than 1 iff $p < 1/2$.

▶ **Lemma 6.7.** Let ℓ, u be integers such that $0 \le \ell \le u \le n$. The expected number of distinct sets in $\mathcal{B}_{n,m,p}$ satisfies

$$\frac{m}{1+m\mathbf{p}} \cdot \mathrm{P}[\ell \le X \le u] \le \mathrm{E}\big[\|\mathcal{B}_{n,m,p}(\ell,u)\|\big] \le m \cdot \mathrm{P}[\ell \le X \le u]. \qquad \blacktriangleleft$$

Proof. We apply Proposition 2.12 to prove both the upper and the lower bound.

$$\mathrm{E}\big[\,\|\mathcal{B}_{n,m,p}(\ell,u)\|\,\big] = \sum_{i=\ell}^{u} \binom{n}{i} (1-(1-p^i(1-p)^{n-i})^m) \le m \cdot \sum_{i=\ell}^{u} \binom{n}{i} p^i(1-p)^{n-i}.$$

$$\mathrm{E}\big[\,\|\mathcal{B}_{n,m,p}(\ell,u)\|\,\big] \ge \sum_{i=\ell}^{u} \binom{n}{i} \frac{m p^i(1-p)^{n-i}}{1+m p^i(1-p)^{n-i}} \ge \frac{m}{1+m\mathbf{p}} \cdot \sum_{i=\ell}^{u} \binom{n}{i} p^i(1-p)^{n-i}.$$

■

6.4 The Size of the Minimization

We now prove the main results about the number of minimal edges of $\mathcal{B}_{n,m,p}$. The key observation is that the minimization is dominated by the sets whose cardinality is at the threshold $i^* = n + \log_{1-p} m = (1-\alpha)n$.

6.4.1 Binomial Characterization

Theorem 6.1 establishes a close connection between the expected size of the minimization and the binomial distribution. We divide its proof into three lemmas corresponding, in that order, to the lower bound in the first statement of the theorem, the tight upper bound, and the second statement.

Note that the following lemma is slightly more general than what is claimed in Theorem 6.1 (*i*). It pertains to all $m \le 1/(1-p)^n$, in other words, it does not require $\alpha = \alpha(m,n)$ to be bounded away from 1.

▶ **Lemma 6.8 (lower bound of Theorem 6.1 (*i*)).** For all $m \le 1/(1-p)^n$, that is, all $0 \le \alpha \le 1$, it holds that $\mathrm{E}\big[|\min(\mathcal{B}_{n,m,p})|\big] = \Omega(m) \cdot \mathrm{P}[X \le i^*]$. ◀

Proof. If $m = 1$, then $i^* = n$ and $|\min(\mathcal{B}_{n,m,p})| = 1 = m \cdot \mathrm{P}[X \le i^*]$ holds deterministically. Hence, we assume $m > 1$. The sought expectation is at least as large as the number of distinct minimal sets up to some cardinality i, for an arbitrary choice of $i \le n$. As an ansatz, we set it equal to the threshold $i^* = n + \log_{1-p} m = (1-\alpha)n$. Without loosing generality, i^* is an integer; otherwise, we take $\lfloor i^* \rfloor$. By the assumption $m \le 1/(1-p)^n$, we have $i^* \ge 0$. Let $\mathbf{p} = \max_{0 \le i \le i^*}\{p^i(1-p)^{n-i}\}$. Lemma 6.4 (*i*) and 6.7 imply that $\mathrm{E}\big[|\min(\mathcal{B}_{n,m,p})|\big]$ is lower-bounded by

$$\begin{aligned}\sum_{i=0}^{i^*} \binom{n}{i} s_{n,p}(i,m) \cdot w_{n,p}(i,m) &\ge \sum_{i=0}^{i^*} \binom{n}{i} s_{n,p}(i,m) \cdot w_{n,p}(i^*,m) \\ &= \mathrm{E}\big[\|\mathcal{B}_{n,m,p}(0,i^*)\|\big] \cdot w_{n,p}(i^*,m) \ge \frac{1}{1+m\mathbf{p}} \cdot w_{n,p}(i^*,m) \cdot m \cdot \mathrm{P}[X \le i^*].\end{aligned}$$

To complete the proof, we verify that neither $1/(1+m\mathbf{p})$ nor $w_{n,p}(i^*,m)$ vanish for any $m \le 1/(1-p)^n$. For the first factor, this is immediate from the closed form for $\mathbf{p}$ (see the paragraph preceding Lemma 6.7). It implies $m\mathbf{p} = \max_{0 \le i \le i^*}\{mp^i(1-p)^{n-i}\} \le \max\{1, p^{i^*}\} = 1$. Corollary 6.6 shows that $w_{n,p}(i^*,m)$ is bounded away from 0 universally for all $1 < m \le 1/(1-p)^n$. ■

▶ **Lemma 6.9 (upper bound of Theorem 6.1 (*i*)).** For any constant $\varepsilon > 0$ and all $m \le 1/(1-p)^{(1-\varepsilon)n}$, that is, all $0 \le \alpha \le 1-\varepsilon$, it holds that $\mathrm{E}\big[|\min(\mathcal{B}_{n,m,p})|\big] = \mathrm{O}(m) \cdot \mathrm{P}[X \le i^*]$. The leading constant may depend on ε. ◀

Proof. By Lemma 6.4, we know that

$$\mathrm{E}\big[|\min(\mathcal{B}_{n,m,p})|\big] \le \sum_{i=0}^{n} \binom{n}{i} s_{n,p}(i,m) \cdot w_{n,p}(i,m-1)$$

We split the sum at the threshold $i^* = (1-\alpha)n$ and handle the two parts separately.

Regarding the first part with indices up to i^*, Lemma 6.7 implies

$$\sum_{i=0}^{i^*} \binom{n}{i} s_{n,p}(i,m) \cdot w_{n,p}(i,m-1) \leq \sum_{i=0}^{i^*} \binom{n}{i} s_{n,p}(i,m) \\ = \mathrm{E}\big[\, \|\mathcal{B}_{n,m,p}(0,i^*)\| \,\big] \leq m \cdot \mathrm{P}[X \leq i^*].$$

For larger indices, we can no longer ignore the influence of the weighting factors. We show that the whole second part of the sum is only a constant factor larger than $m\mathrm{P}[X = i^*]$ (and therefore in $\mathrm{O}(m)\mathrm{P}[X \leq i^*]$). Let $\ell \leq n - i^*$ be a positive integer and consider the $(i^*+\ell)$-th term of the sum

$$t_{n,m,p}(i^*+\ell) = \binom{n}{i^*+\ell} s_{n,p}(i^*+\ell, m) \cdot w_{n,p}(i^*+\ell, m-1).$$

If $\ell = n - i^*$ the term contributes at most 1. For the remainder, assume $\ell < n - i^*$. Using Proposition 2.12 the same way as in the proof of Lemma 6.7 shows that $t_{n,m,p}(i^*+\ell) \leq m\mathrm{P}[X = i^*+\ell] \cdot w_{n,p}(i^*+\ell, m-1)$. For the ratio between the term and $m\mathrm{P}[X = i^*]$ we thus get

$$r(\ell) = \frac{t_{n,m,p}(i^*+\ell)}{m\mathrm{P}[X = i^*]} \leq \frac{m\mathrm{P}[X = i^*+\ell]}{m\mathrm{P}[X = i^*]} \cdot w_{n,p}(i^*+\ell, m-1) \\ = \frac{\binom{n}{i^*+\ell}}{\binom{n}{i^*}} \left(\frac{p}{1-p}\right)^{\ell} w_{n,p}(i^*+\ell, m-1).$$

We show that the first factor is bounded using that $\alpha \leq 1 - \varepsilon$.

$$\frac{\binom{n}{i^*+\ell}}{\binom{n}{i^*}} = \prod_{j=1}^{\ell} \frac{n - i^* - j + 1}{i^* + j} \leq \left(\frac{n - i^*}{i^*}\right)^{\ell} = \left(\frac{\alpha}{1-\alpha}\right)^{\ell} \leq \frac{1}{\varepsilon^{\ell}}.$$

The assumption $\ell < n - i^*$ ensures that the ratio between subsequent weighting factors is a constant. In more detail, we have proven in Lemma 6.4 *(iii)* that $w_{n,p}(i^*+\ell, m-1)/w_{n,p}(i^*+\ell, m) \leq 1/p$. The same assumption also makes Lemma 6.5 *(i)* applicable. Combining the two facts gives the following.

$$w_{n,p}(i^*+\ell, m-1) = \frac{w_{n,p}(i^*+\ell, m-1)}{w_{n,p}(i^*+\ell, m)} \cdot w_{n,p}(i^*+\ell, m)$$
$$\leq \frac{1}{p}\exp(-m(1-p)^{n-i^*-\ell+1}) = \frac{1}{p}\exp(-(1-p)^{-\ell+1}).$$

Define $a = p/\varepsilon(1-p)$ and $b = 1/(1-p)$. So far, we have established that the ratio between the $(i^*+\ell)$-th term and $m\mathrm{P}[X = i^*]$ is at most

$$r(\ell) \leq \frac{1}{p}\frac{a^\ell}{\exp(b^{\ell-1})}.$$

The bound on r itself is already free of any dependence on n, m, or α. However, in order to prove our claim, we need to bound the sum $1 + \sum_{\ell=1}^{n-i^*-1} r(\ell)$. A fortiori, we show that the series $\sum_{\ell=1}^{\infty} r(\ell)$ is summable. To this end, consider the sequence $q(\ell) = r(\ell) \cdot 2^\ell$. Its logarithm $\ln q = \ell \ln(2a) - \ln p - b^{\ell-1}$ diverges to $-\infty$ as ℓ increases, implying $q \to 0$. This means, there exists an ℓ_0 such that $r(\ell) \leq 2^{-\ell}$ for all $\ell \geq \ell_0$.

$$\sum_{\ell=1}^{\infty} r(\ell) \leq \sum_{\ell=1}^{\ell_0} \frac{a^\ell}{p \cdot \exp(b^{\ell-1})} + \sum_{\ell=\ell_0+1}^{\infty} \frac{1}{2^\ell} \leq \sum_{\ell=1}^{\ell_0} \frac{a^\ell}{p \cdot \exp(b^{\ell-1})} + 2 = \mathrm{O}(1). \quad \blacksquare$$

We have characterized $\mathrm{E}\big[|\min(\mathcal{B}_{n,m,p})|\big]$ in terms of the binomial distribution as long as $m \leq 1/(1-p)^{(1-\varepsilon)n}$. Also, $|\min(\mathcal{B}_{n,m,p})|$ always contains at least one edge by definition. We now prove that once m is a polynomial factor (in n) larger than $1/(1-p)^n$ this trivial lower bound is tight up to lower order terms.

▶ **Lemma 6.10 (Theorem 6.1 (*ii*)).** There is a constant $c > 0$ such that for all $m \geq 1/(1-p)^{n+c\ln n}$, i.e., $\alpha \geq 1 + \frac{c\ln n}{n}$, we have $\mathrm{E}\big[|\min(\mathcal{B}_{n,m,p})|\big] = 1 + \mathrm{o}(1)$. ◀

Proof. Let $m = 1/(1-p)^{n+f}$ for some non-negative function $f = f(n)$. As soon as the empty set is sampled in one of the m trials, the minimization of $\mathcal{B}_{n,m,p}$ comprises only a single set; otherwise, we fall back to the trivial estimate $|\min(\mathcal{B}_{n,m,p})| \leq m$. Let A denote the event $[\emptyset \in \mathcal{B}_{n,m,p}]$. The law of total expectation together with Corollary 2.11 implies

$$\mathrm{E}\big[|\min(\mathcal{B}_{n,m,p})|\big] = \mathrm{E}\big[|\min(\mathcal{B}_{n,m,p})| \,\big|\, A\big]\mathrm{P}[A] + \mathrm{E}\big[|\min(\mathcal{B}_{n,m,p})| \,\big|\, \neg A\big]\mathrm{P}[\neg A]$$

$$\leq \mathrm{P}[A] + m \cdot (1-(1-p)^n)^m$$
$$\leq 1 + \exp(\ln m - m(1-p)^n) = 1 + \exp\Big(\ln m - (1-p)^{-f}\Big).$$

We have $\ln m = -\ln(1-p)(n+f)$. Note that $-\ln(1-p)$ is a constant strictly larger than 1. Requiring $f \geq (c' \ln n)/(-\ln(1-p))$ for an arbitrary constant $c' > 1$ ensures that $\ln m$ is negligible compared to $(1-p)^{-f}$, whence the expression above converges to 1. Setting $c = -c'/\ln(1-p)$ gives the lemma. ■

6.4.2 The Case $m = 1/(1-p)^n$

There is a mismatch between the upper and lower bounds above in the range of parameters for which they hold. The lower bound has been proven for all $m \leq 1/(1-p)^n$. In particular, this includes the case where $\alpha = \alpha(n)$ converges to 1 from below. Consider any α with $1 - 1/n < \alpha \leq 1$. This means that $\mathrm{P}[X \leq (1-\alpha)n] = \mathrm{P}[X = 0] = (1-p)^n$ and $1/(1-p)^{n-1} < m \leq 1/(1-p)^n$. Inserting this in Lemma 6.8 gives $\mathrm{E}\big[|\min(\mathcal{B}_{n,m,p})|\big] = \Omega(m) \cdot \mathrm{P}[X \leq (1-\alpha)n] = \Omega(1) \cdot (1-p) = \Omega(1)$. This is not surprising as we always have $|\min(\mathcal{B}_{n,m,p})| \geq 1$.

On the other hand, the upper bound in Lemma 6.9 has been proven only for $\alpha \leq 1-\varepsilon$ for any constant $\varepsilon > 0$. If we still were to insert some $1-1/n < \alpha \leq 1$, we would get $\mathrm{E}\big[|\min(\mathcal{B}_{n,m,p})|\big] = \mathrm{O}(1)$. We refute this estimate below. Contrarily, we prove a lower bound for the case $\alpha > 1 - 1/n$ that is polynomially larger than the one inherited from Lemma 6.8. This shows that the binomial characterization of Theorem 6.1 breaks down if α converges to 1 (sufficiently fast).

▶ **Lemma 6.11.** If $1 - 1/n \leq \alpha \leq 1$, that is $1/(1-p)^{n-1} \leq m \leq 1/(1-p)^n$, then we have $\mathrm{E}\big[|\min(\mathcal{B}_{n,m,p})|\big] = \Omega(n)$. ◀

Proof. We extend the proof idea of Lemma 6.10. If none of the m trials generates the empty set, then all distinct sampled singletons are minimal.

$$\mathrm{E}\big[|\min(\mathcal{B}_{n,m,p})|\big] \geq \mathrm{E}\big[\|\mathcal{B}_{n,m,p}(1,1)\|\big] \cdot \mathrm{P}\big[\emptyset \notin \mathcal{B}_{n,m,p}\big]$$
$$= n \cdot s_{n,p}(1,m) \cdot (1 - s_{n,p}(0,m)).$$

We show that the product $s_{n,p}(1,m)\,(1 - s_{n,p}(0,m))$ is bounded away from 0 for all n using Proposition 2.12 and our assumption on the range of m.

$$s_{n,p}(1,m) = 1-(1-p(1-p)^{n-1})^m \geq \frac{mp(1-p)^{n-1}}{1+mp(1-p)^{n-1}} \geq \frac{p}{1+p/(1-p)} = p(1-p).$$

For the second factor, we use Corollary 2.11.

$$1-s_{n,p}(0,m) = (1-(1-p)^n)^m \geq \exp(-m(1-p)^n)\cdot(1-m(1-p)^{2n}) \geq \exp(-1)\cdot(1-(1-p)^n) \geq \frac{p}{e}.$$ ■

Comparing the Lemmas 6.10 and 6.11 shows that a slight polynomial increase (in n) of the number of trials beyond $1/(1-p)^n$ is enough to push the size of the minimization from $\Omega(n)$ to $1+o(1)$. We leave it as an open problem to give exact bounds for this collapse around $m \approx 1/(1-p)^n$, that is, $\alpha \approx 1$.

Observe that in our illustration above the choice of α is tight in the sense that for $\alpha = 1-1/n$, we have $\mathrm{P}[X \leq (1-\alpha)n] = \mathrm{P}[X \leq 1] = (1-p)^n + np(1-p)^{n-1}$. For the corresponding $m = 1/(1-p)^{n-1}$, Lemma 6.8 implies $\mathrm{E}\left[|\min(\mathcal{B}_{n,m,p})|\right] = O(n)$ and Lemma 6.11 proves a *matching* lower bound. The difference stems from the fact that the binomial variable X is discrete and thus $\mathrm{P}[X \leq (1-\alpha)n]$ as a function of n introduces some quantization error. However, we do not know whether this is the only obstacle for extending Theorem 6.1.

6.4.3 Phase Transition at $m^* = 1/(1-p)^{(1-p)n}$

We now discuss the behavior of $\min(\mathcal{B}_{n,m,p})$ at $m^* = 1/(1-p)^{(1-p)n}$. We show that the size of the minimization undergoes a phase transition there. This is made explicit in Theorem 6.2 (restated below) and illustrated in Figure 6.1. Intuitively, for a small number of edges, the ratio of minimal edges among all edges is constant and thus $|\min(\mathcal{B}_{n,m,p})|$ scales linearly with m. At the transition point, both the size of the multi-hypergraph as well as its minimization are of order $m^* = 1/(1-p)^{(1-p)n} = 2^{(\mathrm{H}(1-p)+p\,\mathrm{ld}\,p)\,n}$. Here, H again denotes the binary entropy function. This overlap is indicated in Figure 6.1 (b) by dashed lines. Beyond that, in the information-theoretic regime, the size of the minimization is instead given by the perplexity of α, and follows $2^{(\mathrm{H}(\alpha)+(1-\alpha)\,\mathrm{ld}\,p)\,n}$ up to polynomial factors. This means that the growth of the minimization continues at

first, but is now only sublinear. Its size peaks at a certain m, which we compute exactly in Lemma 6.12. For an even larger number of trials, the number of minimal edges begin to *fall*, even exponentially. Once m exceeds $1/(1-p)^n$, the minimization collapses under the sheer likelihood of the empty set being sampled, see Lemma 6.10.

To gauge the phase transition more accurately, we reuse the correction factor φ from Chapter 5. Recall that we defined $\varphi(n,p,p) = 1$ and, for all reals $x \in [0,p)$,

$$\varphi(n,p,x) = \min\left(1, \frac{1}{(p-x)\sqrt{xn}}\right).$$

Let again D denote the (binary Kullback–Leibler) divergence between Bernoulli distributions. We proved in Theorem 5.1 that

$$\mathrm{P}[X \le (1-\alpha)n] = \Theta(1) \cdot \varphi(n,p,1-\alpha) \cdot 2^{-\mathrm{D}(1-\alpha \,\|\, p)n}.$$

Close to the transition point $\alpha \approx 1-p$, we have $\varphi \approx 1$. As α moves further away, φ is shrinking. Once there is a constant additive gap between α and $1-p$ the correction factor is of order $\Theta(1/\sqrt{n})$.

▶ **Theorem 6.2 (restated).**

(*i*) If $m \le 1/(1-p)^{(1-p)n}$, that is $\alpha \le 1-p$, then $\mathrm{E}\big[|\min(\mathcal{B}_{n,m,p})|\big] = \Theta(m)$.

(*ii*) Let $\varepsilon > 0$ be any constant. If m is between $1/(1-p)^{(1-p)n}$ and $1/(1-p)^{(1-\varepsilon)n}$, that is, $1-p \le \alpha \le 1-\varepsilon$, then $\mathrm{E}\big[|\min(\mathcal{B}_{n,m,p})|\big]$ is of order

$$\Theta(1) \cdot \varphi(n,p,1-\alpha) \cdot 2^{(\mathrm{H}(\alpha)+(1-\alpha)\log_2 p)\,n}$$
$$= \Theta(1) \cdot \min\left(1, \frac{1}{(\alpha-(1-p))\sqrt{(1-\alpha)n}}\right) \cdot \left(\frac{p^{1-\alpha}}{(1-\alpha)^{1-\alpha}\,\alpha^{\alpha}}\right)^n. \quad ◀$$

Proof. The first statement covers the linear regime of $m \le 1/(1-p)^{(1-p)n}$. By the binomial characterization in Theorem 6.1, we have $\mathrm{E}\big[|\min(\mathcal{B}_{n,m,p})|\big] = \Theta(m) \cdot \mathrm{P}[X \le (1-\alpha)n]$, with $X \sim \mathrm{Bin}(n,p)$. It is thus enough to verify that the probability does not converge to 0. This follows easily from $\alpha \le 1-p$ and pn being the median of the binomial distribution.

In the remainder, we treat the information-theoretic regime of all m such that $1/(1-p)^{(1-p)} \le m \le 1/(1-p)^{(1-\varepsilon)n}$ for some fixed $\varepsilon > 0$. In particular, $(1-\alpha)n$

is now smaller than the mean $\mathrm{E}[X]$. Our improved Chernoff–Hoeffding bound in Theorem 5.1 yields $\mathrm{E}\big[\,|\min(\mathcal{B}_{n,m,p})|\,\big] = \Theta(m) \cdot \varphi(n,p,1-\alpha) \cdot 2^{-\mathrm{D}(1-\alpha\,\|\,p)\,n}$. We express the divergence power in terms of the perplexity and use the equalities $m = 1/(1-p)^{\alpha n}$ as well as $\mathrm{H}(1-\alpha) = \mathrm{H}(\alpha)$ to obtain

$$\begin{aligned} m \cdot 2^{-\mathrm{D}(1-\alpha\,\|\,p)\,n} &= \frac{1}{(1-p)^{\alpha n}} \cdot 2^{\mathrm{H}(1-\alpha)n}\, p^{(1-\alpha)n}\, (1-p)^{\alpha n} \\ &= 2^{\mathrm{H}(1-\alpha)n}\, p^{(1-\alpha)n} = 2^{(\mathrm{H}(\alpha)+(1-\alpha)\,\mathrm{ld}\,p)\,n}. \end{aligned}$$

■

The behavior of the minimization for increasing m suggests that there is a sweet spot in the information-theoretic regime where the expected number of minimal edges is maximum. We apply Theorem 6.2 to calculate this maximum.

▶ **Lemma 6.12.** Let $0 < p < 1$. The maximum expected size of the minimization over all m is $\max_m \mathrm{E}\big[\,|\min(\mathcal{B}_{n,m,p})|\,\big] = \Theta\big((1+p)^n/\sqrt{n}\big)$. The maximum is attained at $m = 1/(1-p)^{\frac{n}{1+p}}$, that is, $\alpha = 1/(1+p)$. ◀

Proof. We first verify that that the maximum indeed sits in the information-theoretic regime. We use the fact that $1/(1-p)^{1-p} < 1+p$ holds for all $0 < p < 1$. This can be seen from $1/(1-p)^{1-p}$ being strictly concave on the open unit interval and $1+p$ being its tangent line at position $p = 0$.

Since $|\min(\mathcal{B}_{n,m,p})| \le m$, the sample size $m \le 1/(1-p)^{(1-p)n}$ in the linear regime is too small to lead to the claimed bound of order $(1+p)^n/\mathsf{poly}(n)$. Regarding the information-theoretic regime, observe that $f_p(\alpha) = 2^{\mathrm{H}(\alpha)+(1-\alpha)\,\mathrm{ld}\,p}$ is continuous and concave. It converges to $2^{\mathrm{H}(1-p)+p\,\mathrm{ld}\,p} = 1/(1-p)^{(1-p)}$ as $\alpha \searrow 1-p$ (for any fixed p). Hence, there exists some $\beta > 0$ small enough such that $f_p(1-p+\beta) < 1+p$. Conversely, $f_p(\alpha)$ tends to 1 as $\alpha \nearrow 1$. Let $\gamma > 0$ be such that $f_p(1-\gamma) < 1+p$. In other words, any bound that is exponential in $1+p$ must come from $\alpha \in [1-p+\beta, 1-\gamma]$. We are in the setting of Theorem 6.2 *(ii)*.

Let φ abbreviate $\varphi(n,p,1-\alpha)$. There are constants $C, C' > 0$ such that

$$C \cdot \varphi \cdot 2^{(\mathrm{H}(\alpha)+(1-\alpha)\,\mathrm{ld}\,p)\,n} \le \mathrm{E}\big[\,|\min(\mathcal{B}_{n,m,p})|\,\big] \le C' \cdot \varphi \cdot 2^{(\mathrm{H}(\alpha)+(1-\alpha)\,\mathrm{ld}\,p)\,n}.$$

As α is bounded away from both $1-p$ and 1, we have $\varphi = \Theta(1/\sqrt{n})$ and the hidden constants are independent of α (but possibly depending on p, β, and γ). We are thus left to determine the extremum of the exponential part. Let

$g_p = \log_{(2^n)} f_p = (\operatorname{ld} f_p)/n$ be the exponent of f_p, meaning

$$g_p(\alpha) = \mathrm{H}(\alpha) + (1-\alpha)\operatorname{ld} p = -\alpha \operatorname{ld} \alpha - (1-\alpha)\operatorname{ld}(1-\alpha) + (1-\alpha)\operatorname{ld} p$$

Its derivative

$$\frac{\mathrm{d}}{\mathrm{d}\alpha} g_p(\alpha) = \operatorname{ld}\left(\frac{1-\alpha}{\alpha}\right) - \operatorname{ld} p$$

has a single zero in the interval $[1-p+\beta, 1-\gamma]$ at $\alpha^* = 1/(1+p)$, resulting in an exponent of

$$g_p(\alpha^*) = -\frac{1}{1+p}\operatorname{ld}\left(\frac{1}{1+p}\right) - \frac{p}{1+p}\operatorname{ld}\left(\frac{p}{1+p}\right) + \frac{p}{1+p}\operatorname{ld} p = \operatorname{ld}(1+p). \qquad \blacksquare$$

6.5 Conclusion

We gave tight bounds on the expected number of minimal edges of maximum-entropy n-vertex, m-edge multi-hypergraphs with expected edge size pn. We discovered a phase transition with respect to the total number of edges. As long as m is at most $m^* = 1/(1-p)^{(1-p)n}$, the expected size of the minimization is linear in m. Once the sample size increases beyond m^*, the minimization is instead governed by the entropy of the exponent α such that $m = 1/(1-p)^{\alpha n}$. The minimization continues to grow sublinearly until m reaches $1/(1-p)^{\frac{n}{1+p}}$, from there on, it decays rapidly. Raising m above $1/(1-p)^n$ finally results in only the empty edge being minimal.

This is a first step to understand why large real-world databases regularly produce small hypergraphs of minimal difference sets. Of course, fixing the expected size of a difference set as the only constraint is not yet an accurate model for the instances arising in data profiling. A promising direction is to allow different sample probabilities per vertex as well as dependencies between the elements. We feel that additional dependencies, that is, more redundancy in the sampled edges, exacerbates the trend towards small minimizations that we already saw in the independent model. However, proving this would require an extension of the maximum-entropy model to incorporate additional constraints. The ultimate goal would be to lift the modeling from the level of hypergraphs to that of the databases themselves.

Arguably a more accessible next step for future work is to improve our theoretical understanding of $\mathcal{B}_{n,m,p}$. Theorem 6.1 still needs to be generalized to also

cover the case where α converges to 1. Also, it seems possible to strengthen our results to hold not only in expectation but asymptotically almost surely, meaning with probability to $1 - o(1)$. It is also interesting to investigate hypergraphs for which the sampling probability p is not constant, but tends to 0. This would bring the study of random hypergraphs closer to that of sparse random graphs.

7 Hitting Set Enumeration with Partial Information

We devise a novel kind of algorithm for the discovery of unique column combinations based on partial hypergraphs. It utilizes the fast enumeration times in practice in order to speed up the search for the relevant difference sets. An experimental evaluation shows that our method outperforms the current state of the art.

7.1 Introduction

Keys are among the most fundamental type of constraint in relational database theory. A *key* is a set of attributes whose values uniquely identify every record in a given relational instance. This uniqueness property is necessary to determine entities in the data. Keys serve to query, link, and merge entities. A unique column combination is the observation that in a given relational instance a certain set of attributes S does not contain any duplicate entries. In other words, UCCs describe attribute sets that fulfill the necessary properties of a key, whereby `null` values require special consideration, hence UCCs are natural *key candidates*. In practice, key discovery is a recurring activity as keys are often missing for various reasons: on freshly recorded data, keys may have never been defined, on archived and transmitted data, keys sometimes are lost due to format restrictions, on evolving data, keys may be outdated if constraint enforcement is lacking, and on fused and integrated data, keys often invalidate in the presence of key collisions. Because UCCs are also important for many data management tasks other than key discovery, such as anomaly detection, data integration, data modeling, query optimization, and indexing, the ability to discover them efficiently is crucial.

Data profiling algorithms automatically discover metadata, like unique column combinations, from raw data and provide this metadata to any downstream data management task. Research in this area has led to various UCC discovery algorithms, such as `GORDIAN` [Sis+06], `HCA` [AN11], `DUCC` [Hei+13], `Swan` [AQN14], and `HyUCC` [PN17]. Out of these algorithms, only `HyUCC` can process datasets of multiple gigabytes in size in reasonable time, because it combines the strengths

of all previous algorithms (lattice search, candidate inference, and parallelization). At some point, however, even HyUCC fails to process certain datasets, because the approach needs to maintain an exponentially growing search space in main memory, which eventually either exhausts the available memory or, due to search space maintenance, the execution time.

In this thesis, we have discussed extensively the connection between UCC discovery and the transversal hypergraph problem. Hitting set-based techniques were among the first tried for discovery [DT87; MR87] and still play a role today in the form of row-based algorithms. In Chapter 3, we argued that this is inevitable–in a sense–as the two problems are equivalent under parsimonious reductions. When utilizing hitting sets directly, we have to extract the information about the difference sets of the database. While the asymptotic running time of the preprocessing is only quadratic in the number of records, we saw that this is prohibitive in practice. Our experiments in Chapter 4 have also verified that the actual enumeration succeeds rather quickly.

We propose a novel approach that automatically discovers all unique column combinations of a given relational dataset. The algorithm is based on the connection to hitting set enumeration. The key insight we use to improve over the method in Chapter 4 is that most of the information in the record pairs is redundant. The following strategy finds the relevant information much more efficiently. We start the discovery with knowing only a few row pairs in the database (in the extreme case, we ignore the input alltogether at first). Due to this lack of information, the resulting solution candidates might not actually be UCCs. A validation procedure must therefore check whether the hitting set is a true UCC. If not, the validation additionally provides a reason for its incorrectness, pointing directly to the part of the database that the enumeration subroutine needs to include in order to avoid the mistake. Moreover, if we perform the enumeration in a careful manner, we can prove that all previous decisions of the algorithm would have been made exactly the same, even if it had full information about all row pairs of the database upfront. This way, our approach can include the new information on the fly and simply resume the discovery where it was before. Instead of providing full and probably redundant information about the database to the subroutine, the enumeration decides for itself which information is necessary to make the right decisions. Additionally, we speed up the computation by employing the fastest hitting set enumeration algorithm in paractice, namely, MMCS by Murakami and Uno [MU14].

Alluding to its hitting set nature, we named our algorithm ***H**itting set enumeration with **P**artial **I**nformation and **Valid**ation*, `HPIValid` for short. Our approach adopts techniques from both data profiling and hitting set enumeration bringing together the two research communities. We make the following contributions.

(1) **UCC discovery.** We present `HPIValid`, a novel UCC discovery algorithm based on the hitting set perspective on data profiling.

(2) **Hitting set enumeration with partial information.** We prove that the enumeration algorithm `MMCS` remains correct when run on a partial hypergraph, provided that there is a validation procedure for candidate solutions. We believe that this insight can help to also solve more general tasks like the discovery of functional dependencies or denial constraints.

(3) **Subset closedness.** We introduce the concept of subset closedness of hitting set enumeration algorithms, and prove it to be sufficient for enumeration with partial information to succeed. This makes it easy to replace `MMCS` with a different subset-closed enumeration procedure if needed.

(4) **Sampling.** We propose a robust strategy how to extract the relevant information from the database efficiently in the presence of redundancies.

(5) **Exhaustive evaluation.** We evaluate `HPIValid` on dozens of real-world datasets and compare it to the state of the art in UCC discovery `HyUCC`.

We show that `HPIValid` solves instances previously out of reach for UCC discovery. It is also up to three orders of magnitude faster than a non-parallel implementation of `HyUCC` and has a smaller memory footprint. This stems from `HPIValid`'s tree search with validation, while `HyUCC` traverses the power set of all column combinations and stores the previous solutions. The requirement of subset closedness, however, makes `HPIValid` harder to parallelize, but it results in a much more resource and environmentally friendly run time profile.

Related Work. Due to the importance of keys in the relational data model, much research has been conducted on finding keys in a given relational instance. Early research on key discovery, for example by Fadous and Forsyth [FF75], is in fact almost as old as the relational model itself. Beeri et al. [Bee+84] have shown that deciding whether there exists a key of cardinality less than a given value in a given relational instance is an **NP**-complete problem, and we showed that it is likely not fixed-parameter tractable but **W**[2]-complete

when parameterized by the size (Theorem 3.1). Finding all keys or unique column combinations is computationally even harder. For this reason, only few automatic data profiling algorithms exist that can discover all unique column combinations. The Discover Minimal UCCs is equivalent to the Transversal Hypergraph problem (Theorem 3.3 and [EG95]) and therefore also to frequent itemset mining, or monotone dualization, see [EMG08]. Many data profiling algorithms, including the first UCC discovery algorithms, rely on the hitting sets of certain hypergraphs; the process of deriving complete hypergraphs, however, is a bottleneck in the computation (see Chapter 4 or [MR94].

In modern data profiling, there are two types of algorithms: row and column-based approaches. For a comprehensive overview and detailed discussions of the different approaches, we refer to [Abe+18] and [Man16]. Row-based algorithms, such as `GORDIAN` [Sis+06], advance the initial hitting set idea. They compare all records in the input dataset pair-wise to derive all valid, minimal UCCs. Column-based algorithms, such as `HCA` [AN11], `DUCC` [Hei+13], and `Swan` [AQN14], in contrast, systematically enumerate and test individual UCC candidates while using intermediate results to prune the search space. The algorithms vary mostly in their traversal strategies, which are breadth-first bottom-up for `HCA` and a depth-first random walk for `DUCC`. Both row and column-based approaches have their strengths: record comparisons scale well with the number of attributes, and systematic candidate tests scale well with the number of records. Hybrid algorithms aim to combine both sides. The one proposed in [KLZ16] (for more general uniqueness constraints) also exploits the duality between difference sets and UCCs to mutually grow the available information about the search and solution space. `HyUCC` [PN17] switches between column and row-based parts heuristically whenever the progress of the current part is low. `Hyb` [WLL19] is a hybrid algorithm for the discovery of so-called *embedded uniqueness constraints* (eUCs), an extension of UCCs to incomplete data. It proposes special ideas tailored to incomplete data, but is based on the same discovery approach as `HyUCC`. All three algorithms share the caveat of a large memory footprint, `HyUCC` and `Hyb` need to store all discovered UCCs or eUCs during computation, the one in [KLZ16] additionally tracks their minimal hitting sets.

Our approach `HPIValid` also implements a hybrid strategy, but instead of switching in level transitions, the column-based part of the algorithm enumerates preliminary solutions on partial information. The validation of those solution candidates then points to areas of the database where focused sampling can

reveal new information. We show that this partial information approach succeeds to find all UCCs as if the row-based approach had been applied exhaustively. Similar to [KLZ16], the duality with the hitting sets problem is employed to find new relevant difference sets. However, the tree search with validation allows us to compute them without the need of holding all previous solutions in memory.

In principle, every unique column combination is also a functional dependency that determines all other attributes. For this reason, existing algorithms for FD discovery, like `TANE` [Huh+99], `FastFDs` [WGR01], `HyFD` [PN16], or `SmartFD` [Zhu+19] as well as their scalable implementations [SGI19], also support key discovery. However, FD discovery is more involved and, therefore, slower than finding UCCs. Inferring UCCs from FDs is also non-trivial [Abe+18]. It is therefore important to be able to discover UCCs with a specialized algorithm.

Regarding the transversal hypergraph problem, out of the three independent articles raising that problem [DT87; MR87; Rei87], the first two posed the task in the context of databases as they employed hitting sets in the discovery of multi-column dependencies between attributes. It was shown much later that also every hitting set problem can be translated into the discovery of UCCs in certain dataset, making the two problems equivalent. Hypergraphs stemming from real-world databases are known to be particularly suitable for enumeration. The applications of minimal hitting sets have grown far beyond data profiling to domains such as data mining, bioinformatics and AI. We refer the reader to the surveys [EMG08; GV17] for an overview. `MMCS` [MU14] is currently the fastest hitting set enumeration algorithm on real-world instances, see [GV17].

7.2 Sample and (Don't) Restart

The folklore Observation 2.2 implies a two-step approach for the enumeration of unique column combinations. First, compute the difference sets of all record pairs, and then apply one of the known algorithms for hitting set enumeration. From a worst-case perspective this approach cannot be improved as the transversal hypergraph problem and UCC enumeration are equivalent under parsimonious reductions (Chapter 3). However, the hypergraphs generated from real-world databases are usually well-behaved and allow for efficient enumeration. In fact, to compute all difference sets is the actual bottleneck. This observation contrasts sharply with the theoretical running time bounds, which are exponential for the hitting sets [Fom+08], but polynomial for the generation of difference sets.

The core idea to avoid the first step is the following. We first sample a few record pairs and compute their difference sets. This gives a *partial hypergraph*, which might be missing some edges. Nevertheless, we pretend for now that we already have the correct hypergraph and start the hitting set enumeration. Due to the partial information, the candidate solutions we find are no longer guaranteed to be UCCs. Thus, whenever we find a hitting set, we use *validation* to check whether it is a UCC of the original database. If this check fails, we know that the partial hypergraph is in fact incomplete. In this case, the validation procedure itself provides new row pairs whose difference sets yield a yet unknown edge. In a first–oversimplified–approach, we include the new information and restart the enumeration with the updated hypergraph. This is repeated until the validation certifies that every hitting set we find is indeed a UCC. To prove that this idea can be successful, one needs to show that we obtain the true hypergraph until the algorithm terminates. Intuitively, this follows from the fact that missing edges in the partial hypergraph make the instance less constraint, meaning that the lack of information might lead to some unnecessary hitting sets, which are rejected by the validation, but the true UCCs are already present.

In the remainder of this section, we actually show the much stronger statement that we can even eliminate the need of restarts. The crucial concept in the proofs is minimality: a hitting set is minimal, if no proper subset intersects all edges; a UCC is minimal if it contains no strictly smaller valid UCC. First, we show the *correctness* of our algorithm. If the validation procedure asserts that a minimal hitting set of the partial hypergraph is a UCC, then it must also be a *minimal* UCC. Even though we cannot be sure that we have all relevant information yet, we can already start returning the found solution to the user. Secondly, we show *completeness*, meaning that we found all minimal UCCs once the algorithm terminates. Thirdly, we define what it means for an enumeration algorithm to be *subset closed*. We then show that subset-closed algorithms do not need to be restarted on a failed validation. Instead, they can simply resume the enumeration with an updated partial hypergraph. Finally, we note that the `MMCS` [MU14] algorithm we use to enumerate hitting sets is indeed subset closed.

A detailed description of `HPIValid`'s implementation follows in Section 7.3.

7.2.1 Correctness and Completeness

We start by proving that the restart approach enumerates all minimal UCCs of a given database. To do so, we need two things: first, an effective validation

whether a minimal hitting set for the partial hypergraph is also a minimal UCC, and, secondly, the assertion that once the enumeration does not produce any more new edges, we have indeed found all UCCs. We obtain both as corollaries from a structural result about hypergraphs and their transversals.

Let $\mathcal{H}$ and $\mathcal{G}$ be two hypergraphs. Recall from Section 2.2 that we use $\mathcal{H} \preccurlyeq \mathcal{G}$ to indicate that every edge of $\mathcal{H}$ contains an edge of $\mathcal{G}$. As usual, $\mathrm{Tr}(\mathcal{H})$ denotes the Sperner hypergraph of all minimal hitting sets of $\mathcal{H}$ and $\min(\mathcal{H})$ its minimal edges. For any $\mathcal{H}$, we have $\mathrm{Tr}(\mathrm{Tr}(\mathcal{H})) = \min(\mathcal{H})$. The following lemma states that the preorder $\preccurlyeq$ completely captures the duality between hypergraphs and their transverals. This connection may be of independent interest beyond the correctness of our enumeration algorithm.

▶ **Lemma 7.1.** $\mathcal{H} \preccurlyeq \mathcal{G}$ holds if and only if $\mathrm{Tr}(\mathcal{H}) \succcurlyeq \mathrm{Tr}(\mathcal{G})$ does. ◀

Proof. We show first that $\mathcal{H} \preccurlyeq \mathcal{G}$ implies $\mathrm{Tr}(\mathcal{H}) \succcurlyeq \mathrm{Tr}(\mathcal{G})$. Let $T \in \mathrm{Tr}(\mathcal{G})$ be a minimal hitting set for $\mathcal{G}$. As every edge of $\mathcal{H}$ contains some edge of $\mathcal{G}$, the set T is also a hitting set for $\mathcal{H}$. T may not be minimal in that regard, but it contains some minimal transversal from $\mathrm{Tr}(\mathcal{H})$. $\mathrm{Tr}(\mathcal{H}) \succcurlyeq \mathrm{Tr}(\mathcal{G})$ follows from here.

For the other direction, the very same argument proves that $\mathrm{Tr}(\mathcal{H}) \succcurlyeq \mathrm{Tr}(\mathcal{G})$ implies $\mathrm{Tr}(\mathrm{Tr}(\mathcal{H})) \preccurlyeq \mathrm{Tr}(\mathrm{Tr}(\mathcal{G}))$, this is the same as $\min(\mathcal{H}) \preccurlyeq \min(\mathcal{G})$. The proof is completed by applying the transitivity of the preorder $\preccurlyeq$ and the two simple facts $\mathcal{H} \preccurlyeq \min(\mathcal{H})$ and $\min(\mathcal{G}) \preccurlyeq \mathcal{G}$. ■

For the following discussion, we fix an (arbitrary) input database $\mathfrak{r}$. Let $\mathcal{D}$ be the hypergraph of the minimal difference sets of pairs of records, then $\mathrm{Tr}(\mathcal{D})$ are the minimal UCCs. Let $\mathcal{P}$ be the current partial hypergraph consisting of the difference sets sampled so far. It may contain difference sets that are not globally minimal (or not even minimal in $\mathcal{P}$), whence we may have $\mathcal{P} \not\subseteq \mathcal{D}$. Nevertheless, $\mathcal{P} \preccurlyeq \mathcal{D}$ holds regardless because the difference set of any pair of rows contains a minimal difference set from $\mathcal{D}$. Suppose we have an enumeration algorithm for $\mathrm{Tr}(\mathcal{P})$, the minimal hitting sets of the partial hypergraph. Those are the candidates for our validation procedure.

▶ **Corollary 7.2.** An edge of $\mathrm{Tr}(\mathcal{P})$ is a minimal UCC iff it is any UCC. ◀

Proof. The implication from minimal UCC to any UCC is trivial. For the opposite direction, let $T \in \mathrm{Tr}(\mathcal{P})$ be a hitting set for the partial hypergraph such that T is a UCC of the database. T thus contains some *minimal* UCC $T' \in \mathrm{Tr}(\mathcal{D})$. From

$\mathcal{P} \preccurlyeq \mathcal{D}$ and Lemma 7.1, we get $\mathrm{Tr}(\mathcal{D}) \preccurlyeq \mathrm{Tr}(\mathcal{P})$. There exists some hitting set $E \in \mathrm{Tr}(\mathcal{P})$ such that $E \subseteq T'$. As $\mathrm{Tr}(\mathcal{P})$ is a Sperner hypergraph, we must have $T = E \subseteq T' \subseteq T$. All three sets are the same, T itself is the minimal UCC. ■

▶ **Corollary 7.3.** If the enumeration of $\mathrm{Tr}(\mathcal{P})$ does not produce a new edge, we have found all minimal UCCs of $\mathfrak{r}$. ◀

Proof. If no call to the validation procedure reveals a new unhit edge, we know that $\mathrm{Tr}(\mathcal{P}) \subseteq \mathrm{Tr}(\mathcal{D})$. We show that $\mathrm{Tr}(\mathcal{P}) = \mathrm{Tr}(\mathcal{D})$ follows from this. The preorder $\preccurlyeq$ generalizes set inclusion, whence $\mathrm{Tr}(\mathcal{P}) \subseteq \mathrm{Tr}(\mathcal{D})$ implies $\mathrm{Tr}(\mathcal{P}) \preccurlyeq \mathrm{Tr}(\mathcal{D})$. Also, $\mathcal{P} \preccurlyeq \mathcal{D}$ gives $\mathrm{Tr}(\mathcal{P}) \succcurlyeq \mathrm{Tr}(\mathcal{D})$ via Lemma 7.1. Now $\mathrm{Tr}(\mathcal{P}) = \mathrm{Tr}(\mathcal{D})$ holds due to the antisymmetry of $\preccurlyeq$ on Sperner hypergraphs. ■

7.2.2 Forgoing Restarts

Restarting the enumeration every time we discover a new unhit edge is obviously not ideal. The information we currently have about the input has changed and we must either recheck the previously found solutions all over again or keep them in memory over the different runs, which is highly impractical [PN17]. The main obstacle to simply resuming the work with the updated hypergraph is the risk of overlooking solutions that are minimal UCCs of the database but not minimal hitting sets of the partial hypergraph. An enumeration algorithm may base the decision on how to traverse the search space of column combination solely on the current input, which does not reflect the whole information of the database. Past decisions may therefore lock the algorithm out of regions in which the new update reveals undiscovered solutions. Next, we give a sufficient condition for overcoming these obstacles. Any algorithm that meets the condition can be combined with our sampling scheme to discover UCCs without any restarts.

A hitting set enumeration method can be seen as a means to decide, at least implicitly, for vertex sets whether they are minimal hitting sets or not. We call an algorithm *subset-closed* if this decision is never made for a set before it is made for all of its subsets. Note that this does not mean that the algorithm needs to check every subset explicitly. For example, certifying some minimal solution implicitly also decides the minimality of all its sub- and supersets.

▶ **Lemma 7.4.** Any subset-closed enumeration algorithm combined with a sampling scheme and validation discovers all minimal UCCs without restarts. ◀

Proof. We claim that a subset-closed algorithm does not overlook any minimal UCC, even if it is only revealed by some later update of $\mathcal{P}$. The minimal UCCs are exactly the edges of $\mathrm{Tr}(\mathcal{D})$. To reach a contradiction, assume that the claim is false and let solution $T \in \mathrm{Tr}(\mathcal{D}) \setminus \mathrm{Tr}(\mathcal{P})$ be such that T corresponds to a node in the power set lattice that was discarded in a previous computation step. By construction, $\mathcal{P} \preccurlyeq \mathcal{D}$ holds and Lemma 7.1 implies $\mathrm{Tr}(\mathcal{P}) \succcurlyeq \mathrm{Tr}(\mathcal{D})$. In other words, there exists a hitting set $S \in \mathrm{Tr}(\mathcal{P})$ such that $S \subsetneq T$. The algorithm is subset-closed, thus S was previously found and validated. This is a contradiction to T being a minimal UCC. ∎

There are many algorithms known to enumerate minimal hitting sets [GV17], see also Chapter 4. Currently the fastest algorithm in practice is the *Minimal-To-Maximal Conversion Search* (`MMCS`) by Murakami and Uno [MU14]. It is based on the simple observation that a minimal hitting set $T \in \mathrm{Tr}(\mathcal{H})$ must be *irredundant*[1] in the sense that each $v \in T$ must have a *private edge*[2] $E \in \mathcal{H}$ such that $T \cap E = \{v\}$. In fact, the minimal hitting sets are maximal among the irredundant sets (but not all maximal irredundant sets are hitting sets). `MMCS` generates a search tree of candidate solutions in a depth-first manner by adding vertices one by one as long as the set is irredundant. An example computation can be found in Section 7.3.3. All checks of candidate sets are purely local and the traversal order ensures that prior to any check, all subsets of the current candidate have been tested. One can verify that `MMCS` is indeed a subset-closed algorithm, we can thus apply it in the partial information setting.

7.3 Algorithm Description

We first give a high-level overview of our algorithm *Hitting Set Enumeration with Partial Information and Validation* (`HPIValid`) illustrated in Figure 7.1. The different components are subsequently explained in detail. The execution of `HPIValid` is split into four major phases. During *preprocessing*, the input table is read and brought into a form that is suitable for the later enumeration. The algorithm then starts with building a partial hypergraph by computing difference sets of *sampled* row pairs. For this partial hypergraph, the minimal hitting sets are enumerated using the *tree search* of `MMCS`. Whenever the search finds a new

1 Irredundant sets are also said to satisfy the *minimality condition* [MU14].
2 Private edges are called *critical* in [MU14].

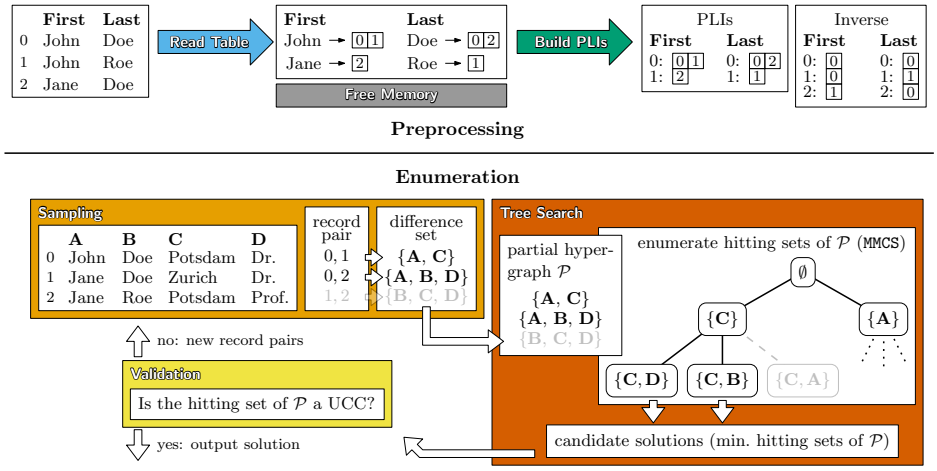


Figure 7.1: Overview of `HPIValid`. During preprocessing, the table is read and the preliminary cluster structure is extracted. The PLIs are created by copying and the memory is subsequently freed. Sampling generates a partial hypergraph of difference sets for the tree search to find new candidate solutions to validate.

candidate solution, the *validation* checks whether the candidate is indeed a UCC. If so, it is a minimal UCC and `HPIValid` outputs it. If the validation shows that a column combination is not unique, we get an explicit list of yet indistinguishable records as a by-product. Every such pair gives a difference set that is not yet contained in the partial hypergraph. We resort again to sampling, focused on those pairs, to manage the influx of new sets. The tree search then resumes with the updated hypergraph from where it was paused.

7.3.1 Preprocessing

The only information from the database that is relevant for the enumeration of its UCCs is the grouping of rows with same value in every column. `HPIValid` uses a data structure called *position list indices*[3] (PLI) to hold this information, as is standard in the literature [AN11; CKS86; Hei+13; Huh+99; PN16; PN17]. Each row is identified by some *record ID*. The PLI of an attribute a is an array of sets of record IDs. It has one set for each distinct value in a and the IDs in any set correspond to exactly those rows that have this value. The sets are called *clusters*

3 Position list indices are also called *stripped partitions* [Huh+99].

and labeled by their *cluster ID*. We say a cluster is *trivial* if it has only a single row; they are not needed for the further computation. A PLI is thus a way to represent the *cluster structure* both of single attributes and column combinations.

The top part of Figure 7.1 shows an example. There are three records with IDs 0, 1, and 2. The values in attribute **First** are ‘‘`John`’’ for record 0 and 1, and ‘‘`Jane`’’ for record 2. The PLI of **First** thus has two clusters. The cluster with ID 0 contains the record IDs 0 and 1. The cluster 1 is trivial and holds only record 2.

The preprocessing has three steps. `HPIValid` first reads the table and, for each column, creates a preliminary hash map from the values to their respective clusters. In the second step, we extract the PLIs from the map. Same values do not necessarily appear in consecutive rows of the input, therefore the memory layout of the clusters is quite ragged after the initial read. To make the later computations more cache efficient, we create the actual PLIs by copying the data from the preliminary map into consecutive memory. Besides the PLIs themselves, we also compute the inverse mappings from record IDs to cluster IDs as in [PN16], see Figure 7.1. This is used in the validation of UCC candidates to efficiently intersect the PLI of a column combination with the PLI of a single attribute; hence, inverse mappings are needed only for single columns.

Finally, we free up the memory occupied by the preliminary data structure. Due to the ragged memory layout, this can take a material amount of time, which we report separately in the evaluation. This step could be avoided trading a higher memory footprint for a slightly smaller run time.

The copying leaves behind some unused memory, neither the mapping from values to clusters nor the initial cluster layout is needed anymore. We are committed to transfer the space efficiency of the hitting set approach into practice as much as possible. Therefore, we free up the memory explicitly at the end of the preprocessing. If one is willing to trade the space reduction for an improved run time, this step can be skipped. For transparency, we report the time to free the memory separately in the evaluation (Section 7.4).

7.3.2 Sampling

Sampling with respect to an attribute a means to draw record pairs coinciding on a uniformly at random (u.a.r.). The PLI of a is a set of clusters $C_1, C_2, \ldots, C_n$ where each C_i contains records with equal values in a. There are $P = \sum_{i=1}^{n} \binom{|C_i|}{2}$

indistinguishable pairs.[4] As P grows quadratically in the cluster size, it is infeasible to compare all pairs by brute force. Instead, we fix some real number x between 0 and 1 as the *sampling exponent* and sample P^x record pairs. As long as there is sampling budget left, we select a cluster i with probability proportional to $\binom{|C_i|}{2}$ and then sample two rows from the cluster u.a.r. without replacement.

The sampling exponent x gives us control over the number of pairs and thus the time needed for the sampling. `HPIValid` allows the user to choose the sampling exponent. Our experiments suggest that $x = 0.3$ is a robust choice, see Section 7.4.1 for a detailed discussion. Comparing the sampled pairs gives the partial hypergraph of difference sets. We need only its inclusion-wise minimal edges for the UCC discovery. We minimize it to save memory.

In the beginning, we sample once with respect to each attribute. The resulting hypergraph $\mathcal{P}$ is potentially missing some edges and we have to resample later. This is done such that every newly sampled record pair is guaranteed to yield an edge not yet included in $\mathcal{P}$. We achieve this by letting the tree search (described in the next section) enumerate minimal hitting sets of $\mathcal{P}$. If the validation concludes that some $S \in \mathrm{Tr}(\mathcal{P})$ is not actually a UCC, then there must be a yet unsampled record pair in the database that coincides on S. Thus, the PLI of S has non-trivial clusters and we sample with respect to S, that is, from all record pairs that are in that cluster. Every difference set found in this way is a witness for the fact that S is not a hitting set of the true hypergraph.

The example in Figure 7.1 has three records. Suppose the initial sampling returns only the pairs $(0, 1)$ and $(0, 2)$ but not $(1, 2)$, that is, $\mathcal{P}$ is missing an edge (grayed out). The tree search will later find the hitting set $S = \{\mathrm{A}\}$. The validation concludes that S is not a UCC and resampling with respect to S yields the missing record pair. The tree search continues with an updated hypergraph.

Our sampling differs from the one of `HyUCC` in that we draw uniformly from a cluster without a pair-picking heuristic like pre-sorting or windowing. A heuristic would have less impact on our algorithm, as we selectively choose only few record pairs from specific clusters, any time we sample.

7.3.3 Tree Search

As mentioned above, we employ `MMCS` [MU14] to enumerate hitting sets of the current partial hypergraph. We can use it almost as a black box, except that we

4 Trivial clusters do not contribute to p since $\binom{1}{2} = 0$.

integrated the validation directly into the search tree to save computation time. We thus briefly discuss the algorithm to an extent necessary for understanding the validation. MMCS on the partial hypergraph $\mathcal{P}$ constructs a search tree of partial solutions. Every node of the tree maintains a set S of vertices (corresponding to attributes in the database) and the collection of those edges $E \in \mathcal{P}$ that are not yet hit by S. Based on a heuristic, MMCS then chooses an unhit edge E^*. By the nature of hitting sets, they contain at least one vertex of E^* and MMCS branches on the decision which $v \in E^*$ to add to S. After branching, the search continues in a child node with the new set $S \cup \{v\}$. If there is no unhit edge left, we have found a hitting set. Beyond the basic branching, further pruning is applied to decrease the search space. A branch is aborted if adding v leads to some vertex of $S \cup \{v\}$ not having a private edge anymore. Also MMCS ensures that no duplicate sets are considered during the enumeration. In particular, before checking some set, all of its subsets must have been checked before.

In Figure 7.1, the partial hypergraph $\mathcal{P}$ consists of two unhit edges. MMCS chooses the smaller $\{A, C\}$ for branching. It selects C in one branch and A in the other. With C selected, $\{A, B, D\}$ is still unhit and the algorithm decides whether to include D, B, or A. The branches with D and B yield minimal hitting sets. MMCS recognizes $S = \{C, A\}$ in the last branch as redundant and prunes.

7.3.4 Validation

As mentioned above, we adapt MMCS to directly integrate the validation. The tree search makes sure that we find the minimal hitting sets of the partial hypergraph. We additionally have to verify that this set is a UCC of the database $\mathfrak{r}$ (see Corollary 7.2). A set $S \cup \{v\}$ of attributes is a UCC if it partitions its subtuples $r[S \cup \{v\}]$, for all records $r \in \mathfrak{r}$, into only trivial clusters. Thus the position list index of $S \cup \{v\}$ is enough to check whether it is a UCC. We obtain this PLI from the one for S and the single column v via PLI intersection, see [Huh+99], where we use the inverse PLIs for optimization as in [PN16]. Suppose that we know the clusters $C_1, \ldots, C_n$ of set S, with trivial clusters already stripped. By intersecting the PLI of S with that of v, the C_i are subdivided into smaller groups corresponding to the same values in column v. For the single-column PLI of v we know the inverse mapping, so for each cluster ID i and every record ID identifying some $r[S] \in C_i$, we look up in which cluster of v this record lies. This gives us a new PLI now representing the cluster structure of $S \cup \{v\}$. Subdivided clusters that became trivial can again be stripped from the partition. Building

these mappings scales with the total number of rows in the C_i. If some set is found to be a hitting set of the partial hypergraph and its cluster structure is empty (contains only trivial clusters), we output it as a minimal UCC; otherwise, there are non-trivial clusters left and we sample new difference sets from these clusters as described in Section 7.3.2.

As some branches of the search tree do not produce hitting sets that have to be validated, it is not necessary to compute the PLIs for every node of the tree. Instead, we intersect the PLIs lazily only along branches that find a solution. To further support the validation, we also extend the branching heuristic of `MMCS`. Recall that the number p counts the indistinguishable pairs in the cluster structure for some column. We transfer this notion to hyperedges by taking the maximum over the values of p for all vertices in the edge. Whenever there are multiple edges of minimum cardinality available, we select E^* among them using the lowest p-maximum as tiebreaker. The idea is that branching on columns with small clusters early reduces the validation cost for all nodes down the path.

7.4 Evaluation

We evaluate the `HPIValid` algorithm with respect to four main aspects.

(1) **Parameter choice.** How should one choose the sampling exponent x? Is there a single universally good choice, independent of the dataset? How robust is the algorithm against small changes in the parameter?

(2) **Performance.** What is `HPIValid`'s run time and memory consumption? How does it compare to the stat-of-the-art solution `HyUCC`?

(3) **Scaling.** How does `HPIValid` scale with the number of rows and columns?

(4) **Reasoning.** Which optimizations contribute to `HPIValid`'s performance?

Experimental setup. `HPIValid` was implemented in C++ and compiled with GCC 10.1.0. We tested it on 54 datasets. Some databases already appeared in Chapter 4 (albeit occasionally with a slightly different number of rows and columns), but most of them are new. The source code and testing data are publicly available.[5] All experiments were run on an Intel® Core™ i7-8700K 3.70 GHz CPU with 32 GB RAM main memory. This is not the same setup as in Section 4.5.1, the CPU is 1.42 times faster but we use 8 times less memory. We

5 hpi.de/friedrich/research/enumdat.html#HPIValid

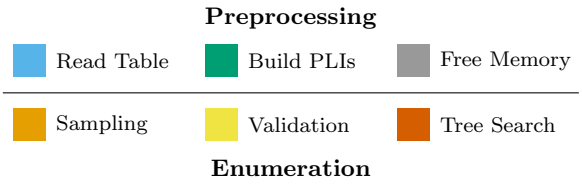


Figure 7.2: Color coding of the run time breakdown.

will see in the evaluation that the speedups in run time cannot be explained by these hardware changes alone. Unless otherwise stated, we used a time limit of 1 h (3600 s). `HyUCC`, implemented in Java, was run with OpenJDK 13 and a heap memory limit of 25 GB. For both, `HPIValid` and `HyUCC`, run times were measured by the algorithms themselves, excluding, e.g., the JVM startup time. Memory consumption was measured using the `time -f '%M'` command. The full results can be found in Tables 7.1 to 7.3, ordered by increasing run time.

Comparability. Run time measurements always also evaluate the implementation. The differences between `HPIValid` and `HyUCC` we observe with respect to run time (Section 7.4.2) and scaling (Section 7.4.3) are beyond what can be explained with implementation details or the difference between C++ and Java. We take the following additional measures to negate the differences. We exclude datasets with very low execution times and use the `-server` flag to optimize Java code more aggressively at run time. We run the compiler and garbage collector in parallel with 4 threads each (`-XX+UseParallelGC -XX:ConcGCThreads=4 -XX:CICompilerCount=4`). Finally, we log the time the Java execution is suspended at safepoints and report the total execution time minus the suspension.

Run time breakdown. We break down the run time of `HPIValid` into the times for specific subtasks. *Read table* is the time to read and parse the input, *build PLIs* refers to the construction of the data structures for the clusters and inverse mapping. The preprocessing is completed by *freeing memory* not needed for the enumeration. See Section 7.3.1 for details. The enumeration phase consists of three tasks that occur interleavingly. *Sampling* and *validation* respectively refer to the time spent to generate new difference sets (Section 7.3.2) and to validate candidate solutions (Section 7.3.4). Everything else is referred to as *tree search*, which is the time required by `MMCS` without the UCC-specific extensions. In the plots below, we associate specific colors with these subtasks, see Figure 7.2 for a legend. (The same colors have already been used in Figure 7.1.)

Table 7.1: Run times and memory consumption of HPIValid and HyUCC (Part 1). All times shown are the median of five runs, except when the algorithm hit the timeout, in which case it is only a single run. The run times for HPIValid are broken down according to the color coding in Figure 7.2. For HyUCC, the reported times exclude the time the JVM suspended execution at safepoints. The actual execution times including this are given in the 'Total' column.

Dataset	Rows [#]	Cols [#]	UCCs [#]	HPIValid Time [s]	HPIValid Time Breakdown	HPIValid Mem [MB]	HyUCC Comparison Total [s]	HyUCC Comparison Time [s]	HyUCC Comparison Mem [MB]	HyUCC Comparison Speedup
horse	300	29	253	1.14 m		8	0.10	0.10	76	92.27
amalgam1_denorm.	51	87	2,737	1.67 m		8	0.07	0.07	77	44.97
t_bioc_measurem.	3.11 k	24	2	3.79 m		9	0.09	0.09	70	22.41
plista	996	63	1	4.67 m		9	0.29	0.29	83	61.39
nursery	13.0 k	9	1	5.74 m		10	0.14	0.14	82	23.54
t_bioc_specim._m.	8.98 k	12	2	8.48 m		11	0.14	0.14	87	16.62
chess	28.1 k	7	1	9.43 m		12	0.15	0.15	100	15.90
letter	18.7 k	17	1	0.02		12	0.43	0.43	132	22.53
flight	1.00 k	109	26,652	0.03		10	1.50	1.48	622	56.02
t_bioc_multimed.	18.8 k	15	4	0.04		27	0.28	0.28	133	7.28
SG_TAXON_NAME	106 k	3	2	0.05		26	0.27	0.27	173	5.57
entytysrcgen	26.1 k	46	3	0.08		31	1.33	1.33	207	17.41
t_bioc_gath_agent	72.7 k	18	4	0.11		41	0.51	0.50	230	4.46
Hospital	115 k	15	12	0.12		48	0.88	0.87	231	7.52
t_bioc_preparation	81.8 k	21	2	0.12		46	0.51	0.50	229	4.19
SPStock	122 k	7	14	0.14		37	0.56	0.55	226	3.93
t_bioc_gath_sitec.	91.3 k	25	2	0.17		56	0.62	0.61	245	3.53
t_bioc_gath_named.	138 k	11	4	0.18		55	0.85	0.83	235	4.62
t_bioc_ident_high.	563 k	3	1	0.18		56	1.08	1.06	374	5.71

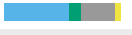
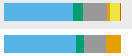
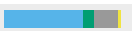
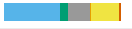
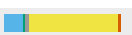
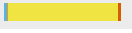
Table 7.2: Run times and memory consumption of HPIValid and HyUCC (Part 2).

[a)]Subset of a larger database with the same name appearing in one of the tables.

Dataset	Rows [#]	Cols [#]	UCCs [#]	HPIValid Time [s]	HPIValid Time Breakdown	HPIValid Mem [MB]	HyUCC Comparison Total [s]	HyUCC Comparison Time [s]	HyUCC Comparison Mem [MB]	HyUCC Comparison Speedup
t_bioc_gath	91.0 k	35	1	0.19		59	1.02	1.00	350	5.36
t_bioc_unit	91.3 k	14	2	0.22		63	0.61	0.59	253	2.66
SG_BIOE._REF_ASS.	358 k	5	1	0.25		66	0.86	0.84	294	3.34
t_bioc_ident	91.8 k	38	2	0.34		82	1.21	1.18	400	3.48
musicbrainz_denorm.	79.6 k	100	2,288	0.45		125	22.24	22.19	1,155	49.22
census	196 k	42	80	0.48		160	394.57	385.19	3,972	800.69
SG_BIOSEQUENCE	184 k	6	1	0.49		143	1.06	1.01	486	2.07
SG_REFERENCE	129 k	6	3	0.51		113	0.67	0.62	341	1.20
SG_BIOENTRY	184 k	9	3	0.52		106	0.78	0.74	308	1.42
SG_BIO._QUAL._ASS.	1.82 M	4	2	0.72		201	3.70	3.62	798	5.02
SG_SEQ._QUAL._ASS.	825 k	4	1	0.76		156	1.25	1.18	486	1.55
SG_BIO._DBXR._ASS.	1.85 M	3	2	0.79		193	2.43	2.37	701	2.98
SG_DBXREF	618 k	4	2	0.89		158	0.88	0.80	513	0.89
SG_SEQFEATURE	1.02 M	6	2	0.97		196	2.19	1.98	894	2.05
ncvoter_allc[a)]	100 k	94	15,244	1.02		193	184.42	184.06	5,920	180.79
Tax	1.00 M	15	13	1.46		214	9.70	9.63	1,352	6.58
SG_LOCATION	1.02 M	8	2	1.76		305	2.26	2.07	1,014	1.17
uniprot[a)]	1.00 k	120	1,973,734	1.93		12	32.47	31.38	8,229	16.24
struct_sheet_range	664 k	32	167	4.14		623	17.89	17.24	2,277	4.17
fd-reduced-30	250 k	30	3,564	4.36		187	115.15	114.89	2,297	26.34
CE4HI01	1.68 M	65	25	6.88		1,514	34.29	33.48	4,674	4.87

Table 7.3: Run times and memory consumption of HPIValid and HyUCC (Part 3). The empty entries for HyUCC indicate that the 25 GB heap memory did not suffice.

[a)]Subset of a larger database with the same name appearing in one of the tables. [b)]Single run with a time limit of 28,800 s (8 h) for HyUCC. [c)]Shrunk from 45 M rows to 20 M to fit into main memory. [d)]'UCCs' column shows the number of solutions HPIValid enumerated within the time limit of 3,600 s (1 h).

Dataset	Rows [#]	Cols [#]	UCCs [#]	HPIValid Time [s]	HPIValid Time Breakdown	HPIValid Mem [MB]	HyUCC Comparison Total [s]	HyUCC Comparison Time [s]	HyUCC Comparison Mem [MB]	HyUCC Comparison Speedup
ZBC00DT_COCM	3.18 M	35	1	9.60		1,752	34.43	32.57	5,233	3.39
isolet[a)]	7.80 k	200	1,282,903	11.12		132	249.74	249.04	2,267	22.40
ditag_feature	3.96 M	13	3	13.09		1,736	109.86	106.41	4,637	8.13
ncvoter_allc[a), b)]	1.50 M	94	206,220	17.12		2,626	>28,800	>28,771	13,701	>1,680
uniprot	539 k	223	826	19.56		2,746				
PDBX_POLY_SEQ_SCH.	17.3 M	13	5	24.40		4,394	83.53	73.52	11,705	3.01
ncvoter	8.06 M	19	96	47.50		4,226	286.48	274.61	10,471	5.78
ILOA[c)]	20.0 M	48	1	48.26		13,208	235.67	217.32	21,599	4.50
VTTS	13.0 M	75	2	60.48		12,373	384.00	373.25	19,098	6.17
lineitem	6.00 M	16	390	81.34		2,234	454.29	448.56	7,172	5.51
ncvoter_allc	7.50 M	94	1,704,511	124.77		12,471	>3,600	>3,549	20,742	>28
tpch_denorm.	6.00 M	52	347,805	2,291.33		9,331	>3,600	>3,586	13,623	>1
uniprot[a), d)]	1.00 k	223	>1,743 M	>3,600		13				
isolet[d)]	7.80 k	618	>153 M	>3,600		295				

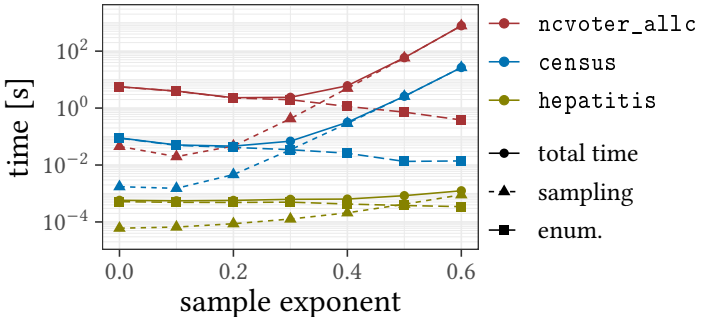


Figure 7.3: Run time scaling of `HPIValid` with respect to the sample exponent x in a log-plot. The time is shown without preprocessing, it is broken down into sampling time and the remaining enumeration. Data points correspond to medians of five runs.

7.4.1 Parameter Choice

An algorithm is only of limited practical value if it requires the user to have prior knowledge which parameter settings work well on their datasets. In our case, we cannot expect the user to know an appropriate value for the sampling exponent x. It is thus crucial that there is either a default setting that works well on many databases or that we can automatically determine a good value for x from the input. Recall that every time we encounter row pairs that are yet indistinguishable with respect to some candidate selection of columns, we compute the number P of such pairs and randomly sample P^x of them for an exponent $0 \leq x \leq 1$. Since P can be quadratic in the number of rows, it makes sense to choose $x \leq 0.5$. Larger values would yield a superlinear running time, which is prohibitively expensive.

In general, a smaller exponent x leads to fewer sampled row pairs, which should be beneficial for the run time. On the other hand, sampling fewer pairs leads to more inaccuracies in the partial hypergraph. It can be assumed that this lack of information misleads the algorithm in the enumeration phase, leading to higher run times. Our experiments confirm this intuition. Figure 7.3 shows run times on three datasets depending on x, divided into sampling time and enumeration time (tree search and validation). As a general trend, the sampling time increases with x, while the time for the remaining enumeration decreases.

In more detail, the sampling time resembles a straight line in the logarithmic plot for larger values of x. This corresponds to an exponential growth in x,

which is to be expected since we sample P^x pairs. The constant or even slightly decreasing sampling time for very small values of x (e.g., between 0 and 0.1 for `census` and `ncvoter_allc` in Figure 7.3) is explained by the fact that we have to sample at least as many difference sets as there are edges in the correct hypergraph. Thus, if we sample fewer pairs, we have to sample more often. The enumeration time (excluding sampling) has a moderate downward trend. Also, for small values of x it is order of magnitudes higher than the sampling time and thus dominates the run time. The situation is reversed for larger values of x. There, the enumeration time decreases only slowly, while the sampling time goes up exponentially, making sampling the dominant factor.

As a result, there is a large range between 0 and 0.4 where the total run time varies only slightly before the sampling time takes over. The minimum appears to be around 0.3. Preliminary experiments confirmed the same effect on other datasets. We conclude that $x = 0.3$ is a good choice for many datasets. Moreover, the fact that 0.3 lies in a wide valley of very similar run times makes it a robust choice. All other experiments in this chapter were done with this setting.

7.4.2 Performance

To evaluate the performance of `HPIValid` and to compare it to `HyUCC`, we ran experiments on 62 datasets of different sizes from various domains. Additionally, we considered truncated variants for some datasets, mainly for consistency with related work [PN17], and to increase the comparability in cases where `HyUCC` exceeded the time limit. Table 7.1 shows the results sorted by run time, excluding thirteen datasets where `HPIValid` took less than 1 ms. For these instances, the speedup factor of `HPIValid` was between 74 and 640.

Run time performance. `HPIValid` solved all but two instances within the time limit of 1 h. The exceptions are `isolet`, and `uniprot` truncated at 1 k rows (but with the full 223 columns). Compared to the other instances, these datasets seem rather special as they have a huge number of UCCs. After 1 h, `HPIValid` enumerated for `isolet` and `uniprot` more than 153 M and 1743 M UCCs, which corresponds to 42 k and 484 k per second, respectively. To prevent I/O from obfuscating the actual run time, we do not output the UCCs. Instead, we merely count the total number of solutions.

Besides the two special cases, only `tpch_denormalized` came close to the time limit with its processing taking 2291 s. All other instances were solved in

less than 3 min. Moreover, the breakdown of the run times in Table 7.1 into the different subtasks (see also Figure 7.2) shows that preprocessing usually makes up more than half of the total run time. The overall performance therefore cannot be significantly improved without improving the preprocessing. There are some interesting exceptions to this trend: On the instances `lineitem`, `ncvoter_allc` and `tpch_denormalized`, `HPIValid` spent the majority of the execution on the validation. We discuss this effect further in the scaling experiments in Section 7.4.3. For the mentioned instances `isolet` and `uniprot` truncated at 1 k rows, the algorithm spent by far the most time with the tree search. The same holds true for these datasets respectively truncated at 200 and 120 columns. Improving upon these run times requires to improve `MMCS`, the state-of-the-art in enumerating minimal hitting sets [GV17; MU14].

Memory consumption. Concerning the consumption of main memory, `HPIValid` tops out at 13 GB, with only three of the datasets requiring more than 10 GB. For those three, already the input size is rather large: the `ncvoter_allc` dataset has 7.5 M rows, `VTTS` has 13 M, and `iloa` 20 M rows. Nine datasets required between 1 GB and 10 GB, and all remaining datasets took less memory. usually significantly so.

Memory consumption compared to `HyUCC`. When comparing `HPIValid` and `HyUCC`, one striking difference is the fact that `HyUCC` has to keep its search front of column combinations in memory. In particular, the front includes all minimal UCCs found so far, while `HPIValid` only needs to store the current branch of the search. It has thus a significantly larger memory footprint, especially on instances with many solutions. In fact, this makes it infeasible to process the two extreme datasets `isolet` and `uniprot` truncated at 1 k rows with `HyUCC`. The variants in which the number of columns are cut at 200 for `isolet` and 120 for `uniprot` can still be solved. However, on the former, `HPIValid` is more memory efficient than `HyUCC` by an order of magnitude. On `uniprot` with 200 columns and 1 k rows, it is by almost three orders of magnitude: `HyUCC` requires 8 GB, `HPIValid` 12 MB.

It is interesting to see that the full `uniprot` dataset with 539 k rows is much more well-behaved than the one truncated at 1 k rows. One could expect that larger databases lead to higher run times, and indeed this is the case for most databases, see Section 7.4.3. However, the `uniprot` dataset is special in that the extended instance has only 826 UCCs and `HPIValid` can solve it in under 20 s using 2.7 GB. We have verified earlier in Chapter 6 that, at least in a random

model, a larger number of rows and therefore more difference sets can indeed result in hypergraphs with fewer and smaller minimal edges. HyUCC on the other hand cannot solve the full uniprot instance due to memory overflow. Thus, the fact that uniprot includes a hard subinstance seems to throw off HyUCC even though the full instance contains only few UCCs. On all remaining instances, HPIValid is also more memory efficient.

Run time compared to HyUCC. HPIValid outperformed HyUCC on every instance with only one exception. On some instances with very few UCCs, HyUCC achieves comparable run times. On many other instances HPIValid was significantly faster than HyUCC. The highest speedup achieved on instances that HyUCC finished was for the census dataset, HPIValid was 800 times faster. On the ncvoter_allc dataset truncated at 1.5 M rows, we ran it with an 8 h timeout, which was exceeded by HyUCC, while HPIValid solved it in below 20 s, a speedup of at least three orders of magnitude.

7.4.3 Scaling

We now evaluate how HPIValid scales with respect to the number of rows and columns. To provide some context, we preface our experiments with a short discussion on worst-case run times from a slightly more theoretical perspective.

Regarding the column scaling, the bottleneck is the actual enumeration (tree search and validation). As there can be exponentially many minimal hitting sets and as many UCCs, the worst-case running time must be exponential [BI95]. However, even if the number of solutions is small, there is no subexponential upper bound known for the MMCS algorithm, which is the core of HPIValid. We thus have to assume that HPIValid scales exponentially with the number of columns, even if the output is small. It is a major open question whether an output-polynomial algorithm exists for the hitting set enumeration problem, see [EMG08]. Concerning the number of rows, in principle, we have to compute the difference set of every record pair, which scales quadratically in the number of rows. Moreover, when building the hypergraph of difference sets, we only keep edges that are minimal, without affecting the solutions. For each new edge we sample, this takes time linear in the number of edges currently in the hypergraph. This means a quadratic run time in the number of difference sets. Thus, the best worst-case upper bound for the run time in terms of the number of rows n is $O(n^4)$. Moreover, there are lower bounds known for the minimization

step based on the Strong Exponential Time Hypothesis (Hypothesis 2.3) implying that there is likely no algorithm with subquadratic running time in n, see the discussion in Section 6.1 of this thesis as well as [BCH16; Gao+18].

Although these worst-case bounds seem to prohibit the solution of large instances, `HPIValid` performs well on practical datasets. The reason for this lies in the fact that these databases behave very differently from worst-case instances. Real-world instances have only comparatively few minimal difference sets [BKN17]. We also observed this in our own experiments in Section 4.5. Indeed, `HPIValid` finds the small difference sets by a focused sampling approach involving only a few record pairs. Similarly, the tree search algorithm `MMCS` has been observed to be fast on hypergraphs arising in practice [GV17; MU14], and the instances emerging from the UCC enumeration problem are no exception to this. The only outliers in our experiments in that regard are `isolet` and `uniprot` truncated at 1 k rows, where the hitting set enumeration is slow, which is not surprising due to their large output size.

Consequently, the goal of our experiments is the gathering of insights into the behavior of `HPIValid` on *typical* datasets rather than worst-case instances. The emphasis of our scaling experiments is on databases other than `isolet`. Nonetheless, we have a short section discussing the column scaling for this as well since `isolet` has by far the most columns among all the tested datasets, and it has been considered before for scaling experiments [PN17]. Although experiments on `isolet` may not reflect the typical scaling, we can still make some interesting observations, in particular on the differences between `HPIValid` and `HyUCC`.

Scaling on the `isolet` dataset. We used the first 40 to 280 columns of `isolet`, stride 40, and ran `HPIValid` and `HyUCC` on the resulting instances. Since the run times for `isolet` are fairly concentrated over the different runs, we used the median of three. We also ran `HPIValid` with a timeout of 15 min for number of columns beyond 280. The results are shown in Figure 7.4.

The run time of `HPIValid` resembles a straight line in the plot with a logarithmic time-axis, meaning an exponential scaling. As discussed before, this is to be expected on certain classes of inputs. In contrast, it is interesting to see that the run time of `HyUCC` as well as the number of UCCs (second plot) seems to scale subexponentially. A possible explanation is that `HPIValid` uses the branching technique of `MMCS`, which potentially scales exponentially, regardless of the output. `HyUCC` on the other hand explores the search space of all column combinations starting with the smaller subsets. Given that the solutions for

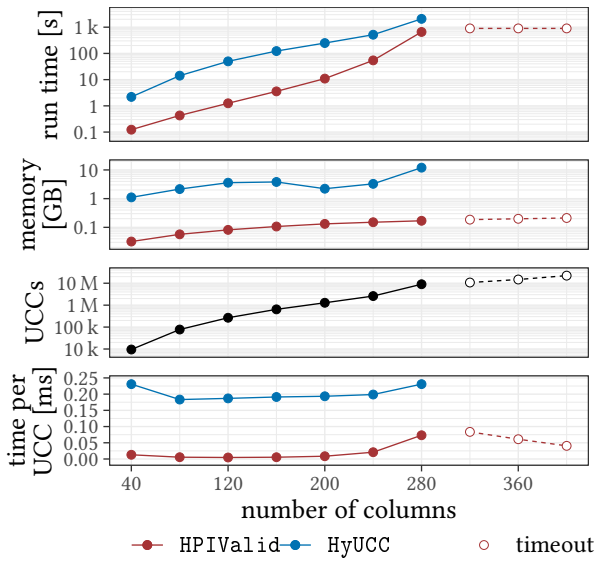


Figure 7.4: Column scaling of `HPIValid` and `HyUCC` on the `isolet` dataset. Shown are the run times, the memory consumption, the number of solutions (log-plots), and the average delay between outputs. Data points correspond to the median of three runs. Dashed lines show runs of `HPIValid` with 15 min timeout.

`isolet` are essentially all column combinations of size 3, then `HyUCC`'s run time is dominated by the output size, which grows cubically here.

Another indicator that the run time of `HPIValid` scales worse than the number of UCCs is the uptick for the time per UCC in the bottom plot of Figure 7.4. The last experiments with timeout effectively provide a snapshot of the first 15 min of execution with many columns. There the trend in the time per UCC is reversed. Although the average delay over the entire run goes up, it is decreasing at the start. This is helpful if one is interested in getting only a few UCCs.

The memory consumption of `HPIValid` on `isolet` shows a similar behavior to the one already discussed in the beginning of Section 7.4.2.

Column scaling on typical instances. We chose two datasets to investigate typical column scaling. The `uniprot` dataset (with all rows) is an obvious choice with its 223 columns. The other dataset, `ncvoter_allc`, is one of the hardest instances for `HPIValid` with a run time of 125 s, and it has enough columns (94) to enable meaningful scaling experiments. Also, `HPIValid` running on

ncvoter_allc spends a substantial amount of time on the enumeration rather than the preprocessing, making this an interesting case study. For additional comparison with HyUCC, we also ran experiments on ncvoter_allc truncated at 100 k rows. The results are shown in Figure 7.5. We discuss them instance by instance from left to right. For the color coding of the run time, recall Figure 7.2.

Truncating ncvoter_allc at 100 k rows makes it small enough so that the preprocessing dominates the run time of HPIValid. It stands out that the run time appears to scale sublinearly with respect to the number of columns. In theory, this cannot happen for an algorithm that processes the whole input at least once. The reason for the observed behavior is that the later columns in the table have fewer different values and more empty cells. This makes reading the table faster as the hash map matching string values to arrays of record IDs needs to cover a smaller domain during dictionary encoding in the preprocessing.

The actual enumeration times are very low with a slight uptick after 50 columns. This is due to the fact that the nature of the instance changes markedly here. The output size increases from a single minimal UCC for the first 50 columns to 4 k solutions at 60 columns. The further increase to 8 k minimal UCCs for 70 columns is not reflected in the enumeration time, starting at 60 columns. Instead, the enumeration time scales linearly in this experiment. The later segment, where the scaling is independent of the output size, shows the more realistic behavior of HPIValid on this dataset. The faster run times on the instances with only a few columns are mere incarnations of the general observation that instances with a single minimal UCC are particularly easy to solve. Every time HPIValid finds a candidate solution for the partial hypergraph, the validation with subsequent sampling produces a new, inclusion-wise smaller difference set, the minimization then effectively replaces the old hypergraph completely. This repeats until there is only a single edge with exactly one vertex left, marking a trivial instance.

The bottom plot shows the scaling behavior of HyUCC in comparison. HyUCC also performs well on instances with a single solution. However, the scaling of HyUCC beyond that point (60+ columns) does not seem to be linear in the number of columns but rather follows the number of UCCs.

The middle column of Figure 7.5 shows the ncvoter_allc dataset with all 7.5 M rows. The output size is also small in the beginning, but there is always more than one solution. The overall column scaling is roughly linear. However, the box plots show that the run time has a high variance, which mainly comes

from the validation. A possible explanation is as follows. Recall that every node in the MMCS search tree corresponds to a set of columns that was selected to be part of the solution. The core operation of the validation is the intersection of the PLI of this subset with the PLI of a single column. The time required for the intersection is linear in the number of rows that are not already unique with respect to the selected columns. If the search happens to select columns high up in the tree that make many rows unique, all intersections in the lower subtrees are sped up. Whether this occurs in any given run of the algorithm heavily depends on the difference sets present in the partial hypergraph when starting the enumeration. As the initial partial hypergraph is random to some extent, the run times vary strongly. The run time breakdowns in the top row of Figure 7.5 show average values and thus give an estimation of the expected running time of our randomized algorithm (in addition to the median values of the runs shown in the box plots).

The comparison with HyUCC on ncvoter_allc with full rows is difficult due to the large run times. The measurements for HyUCC in the bottom plot are thus restricted to inputs with up to 40 columns. The scaling of HyUCC is clearly worse, although there are only few solutions in that range (403 UCCs at 40 columns).

Finally, the right column of Figure 7.5 shows that the run time of HPIValid on uniprot is again dominated by the preprocessing. The enumeration part in turn consists mainly of sampling new difference sets rather than the validation, as for ncvoter_allc. This makes sense in the light of the results in Section 7.4.2. Truncating the dataset at 1 k rows gives a hard subinstance with billions of minimal solutions. It thus cannot suffice to sample only a few record pairs and hope to come close to the correct hypergraph. In other words, redundant information seems to be not as prevalent in uniprot as in other databases. This transfers to the full instance and explains why the algorithm spends much time with computing difference sets. The number of minimal UCCs in the full dataset grows linearly with the number of columns, in contrast to the sudden jump for ncvoter_allc. Possibly the columns of ncvoter_allc represent attributes with distinct characteristics, while those of uniprot seem to be more uniform.

The bottom plot shows that the scaling of HyUCC is superlinear in the number of columns. As observed before, HyUCC appears to have difficulties with uniprot due its hard subinstance. In Section 7.4.2, this became apparent with respect to the memory consumption. Here, it also pertains to the run time. Thus, even with hypothetically unbounded memory, the run time of HyUCC is infeasibly high.

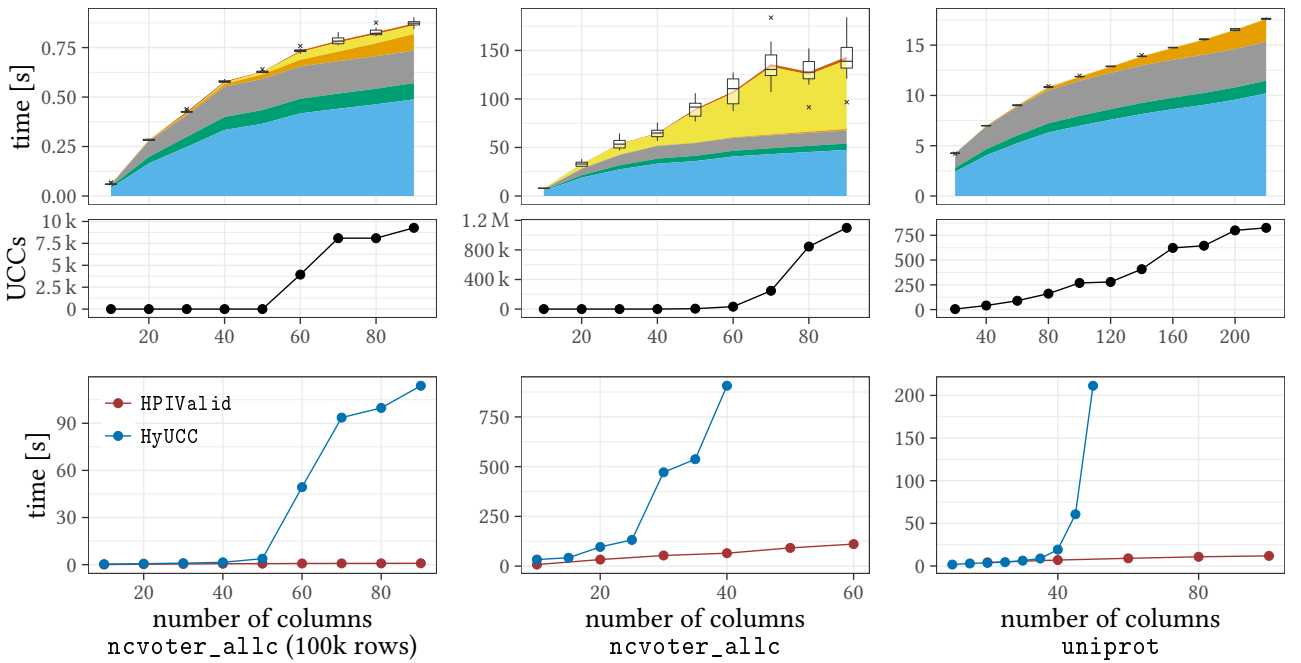


Figure 7.5: Run time scaling of `HPIValid` with respect to the number of columns. Each column shows one dataset. The top row shows the run time averaged over 15 runs of `HPIValid` broken down according to the color coding in Figure 7.2. The middle row shows the number of UCCs. The bottom row compares the median run times of `HPIValid` with that of `HyUCC` with five runs each. Note the different abscissas in the bottom row.

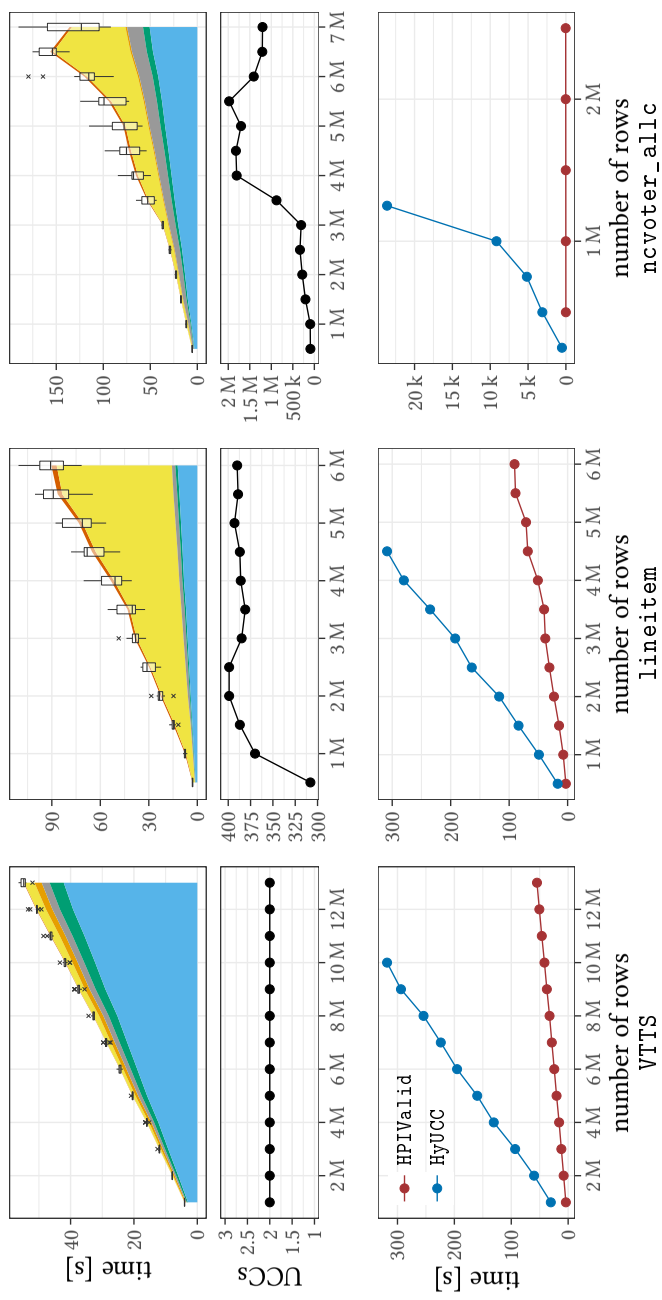


Figure 7.6: Run time scaling of `HPIValid` with respect to the number of rows. Each column represents one dataset. The top row averages over 15 runs of `HPIValid` broken down according to the colors in Figure 7.2. The middle shows the number of UCCs. The bottom row compares the median run times of `HPIValid` with that of `HyUCC` with five runs each for `VTTS` and `lineitem`. Only one `HyUCC` run was done for `ncvoter_allc`, note the different abscissa.

Row scaling on typical instances. For the row scaling experiments, we chose the datasets VTTS, lineitem, and ncvoter_allc. There, HPIValid had the highest run time (apart from tpch_denormalized). They have 13 M, 6 M, and 7.5 M rows, respectively. The results are shown in Figure 7.6. We note that many aspects discussed above for the column scaling apply here as well. We thus focus on the specifics of the row scaling.

VTTS is the largest of the selected datasets, but HPIValid has the lowest run time there. The database carries only two minimal UCCs independently of the number of rows, making the linear preprocessing dominant.

The number of solutions for lineitem is also rather steady but on a higher level. It ranges from 300 to 400 suggesting that also the correct hypergraphs of difference sets vary not too much for the different numbers of rows, given that a critical minimum value is exceeded. This gives a clean straight line for the average run time and most of this time is spent on the validation. For the same reasons already discussed in the context of column scaling above, the validation time has a comparatively high variance as indicated by the box plots.

The output size varies heavily for ncvoter_allc, in stark contrast to the other two datasets. From 500 k to 3 M rows, the number of solutions ranges from 91 k to 335 k; from 3 M to around 4 M rows, there is a jump to 1.8 M minimal UCCs. After this, the output size remains high until it goes down to 1.2 M at 7 M rows. Thus, the hypergraph of difference sets also changes significantly when adding more and more rows. This is reflected in the trend of the average run time being not as clean as the one for lineitem, although for both datasets the execution of HPIValid is dominated by the validation. Additionally, the run time has high variance again, the effect is even more prominent here due to the higher number of columns (lineitem has 16, while there are 94 in ncvoter_allc).

When comparing the row scaling of HPIValid with that of HyUCC, VTTS and lineitem are quite similar. Both algorithms seem to scale linearly, but with HyUCC having a steeper slope. For ncvoter_allc, the scaling is very different (note the different abscissa). HPIValid scales slightly superlinear, while HyUCC shows a sudden jump on instances truncated at more than 1 M. Further scaling experiments became infeasible even with an extended time limit of 8 h

7.4.4 Reasons for Efficiency

There are two crucial factors for the efficiency of HPIValid. First, the number of record pairs for which we compute the difference sets, and secondly, the

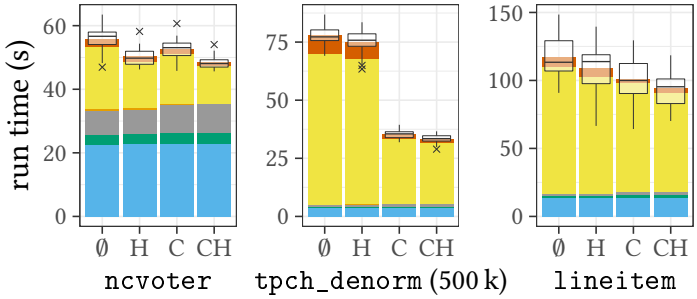


Figure 7.7: Run times with and without tiebreaking heuristic (H) and PLI copying (C) for the datasets `ncvoter`, `tpch_denormalized` (reduced to 500 k rows), and `lineitem`. Each column is based on 25 runs.

size of the search tree. Concerning the number of difference sets, we initially sample $P^{0.3}$ record pairs where P is the total number of available pairs. This is sublinear in the number of records and thus always fast. However, the resulting hypergraph can be incomplete, which forces us to resample additional pairs later on. We measure this using the *relative resampling rate*, which is the number of difference sets computed after the initial sampling divided by the number of difference sets computed in the initial sampling. Excluding two outliers, all instances considered in Section 7.4.2 have a relative resampling rate below 0.36, with a median of 0.00038 and a mean of 0.041. The outliers are the two `isolet` instances with different numbers of columns. They had rates of 15.1 and 10.0.

Concerning the search tree, the number of leaves is at least the number of solutions. In the best case, the tree consists of just the root together with one child per solution. To measure the efficiency of the tree search, we consider the *solution overhead*, that is, the number of non-root nodes per solution. For all instances with at least two solutions, the maximum overhead was 4.5, with a median of 1.4 and mean 1.7. This shows that the `MMCS` tree search enumerates hitting sets very efficiently even when working with partial information. On instances with only a single solution, we got a maximum overhead of 54, with a median of 5.0 and mean 10.5. Although these numbers are higher, they still indicate small search trees. Note that, due to having only a single solution, the numbers here actually represent the absolute tree sizes.

The efficiency of `HPIValid` is mainly due to the aspects discussed above. Beyond that, we have two minor optimizations that slightly improve the run

time. First, to make the validation more cache efficient, we copy the PLIs such that each PLI lies in a consecutive memory block. Secondly, whenever `MMCS` has multiple edges of minimum cardinality to branch on, we use a tiebreaker that aims at speeding up the validation by making the clusters in the resulting PLIs small. For this, we rank the columns by *uniqueness*, with a lower number of indistinguishable record pairs with respect to that column meaning higher uniqueness. Among the edges with minimum cardinality, we choose the one where the least unique column is most unique.

On instances where the preprocessing dominates the run time–see the breakdown in Table 7.1–the effect of these two optimizations is negligible. However, one can see in Figure 7.7 that both optimizations improve the validation time with the caveat that the PLI copying slightly increases the preprocessing time.

7.5 Conclusion

We proposed a novel approach for the discovery of unique column combinations. It is based on new insights into the hitting set enumeration problem with partial information, where the lack of relevant edges is compensated by a validation procedure. Our evaluation showed that the resulting `HPIValid` algorithm outperforms the current state of the art. On most instances, the enumeration times are so small that they are dominated by the preprocessing. This indicates that the room for further improvements is somewhat limited. We believe that it is much more promising to study how our new techniques can be used to solve other problems. Embedded uniqueness constraints, for example, are a generalization of UCCs to incomplete datasets with a similar discovery process [WLL19]. Also closely related are functional dependencies: one can transform their discovery into a hitting set problem as well, only with more and slightly different hypergraphs, see Theorem 3.3. Same as for UCCs, the direct translation is infeasible in practice due to the quadratic number of record pairs, but it seems that some hybrid discovery algorithms for FDs and eUCs [PN16; WLL19] could be accelerated by enumeration with partial information. Our approach may even be able to tackle the much more complex discovery problem for denial constraints, which are a broad generalization of functional dependencies [BKN17; Liv+20].

8 Conclusion & Outlook

Although the relation between data profiling and the enumeration of hitting sets has been known since its inception, we showed here that this perspective still offers an avenue for significant progress. In the complexity-theoretic part of this work, we proved that the two areas are even more intricately connected than previously thought. The discovery problems for several types of multi-column depedencies have equivalent formulations as hitting set problems or, in the case of inclusion dependencies, as maximal *non*-hitting sets. This approach further allowed us to give an enumeration algorithm that is particularly efficient when applied to instances arising in data profiling. Namely, whenever the solution size is bounded, our method has a polynomial worst-case delay that is much smaller than the total running time. Due to its generality, the algorithm may also be useful in other areas where Transversal Hypergraph appears as a subproblem, like artificial intelligence and bioinfomatics. The particular challenge in data profiling was the slow (albeit polynomial) translation of databases to hypergraphs, compared to the fast (but potentially exponential) enumeration. We solved this by moving the two parts even closer together so that they eventually formed interleaving phases of the same algorithm. The result, `HPIValid`, is currently the most efficient discovery method for unique column combinations.

We had to leave some questions in this thesis unanswered. It would yield new insights in enumeration complexity if the Discover Minimal FDs problem, or equivalently Transversal Hypergraph Union, were to admit a parsimonious reduction to Discover Minimal UCCs/Transversal Hypergraph; the insights would be even greater if one could provably rule out such a reduction. Similarly, extending Theorem 6.2 about the minimization of random hypergraphs to the case of vanishing sampling probabilities $p \to 0$ would likely reveal more structural results for sparse hypergraphs. However, we believe that there are much more urgent open problems. Namely, those whose solution further broadens the applicability of data profiling methods and those that bring us closer to resolving *the* most important question in enumeration: whether Transversal Hypergraph has an output-polynomial algorithm.

HPIValid may have a role to play for both aspects. From an engineering standpoint, it is interesting to see whether it can be adapted to also discover other types of multi-column dependencies efficiently. The hope would be that a small modification allows us to find functional dependencies faster than merely rerunning the current algorithm on a slightly different hitting set problem for each possible right-hand side. It may even be able to discover more general *denial constraints*, see [BKN17; Liv+20]. As an illustrating example, imagine a database that lists regular customers as well as staff members that are entitled to employee discounts. Functional dependencies can ensure that there are no two entries with the same Name and Entry Date, but different Account Status. Denial constraints, on the other hand, can model complex requirements like records with a later Entry Date having a higher Customer ID unless the Account Status of the one entry is "customer" and the other is "employee".

On the theoretical side, HPIValid could also help to chart the border of tractability for the Transversal Hypergraph problem. Notably, the algorithm currently comes without any performance guarantees, the same holds for the pure tree search in form of MMCS. We cannot exclude the possibility that both methods are indeed output-polynomial. Admittedly more plausible, however, is the existence of a family of instances on which the overhead of the tree search is super-polynomial. In fact, our empirical analysis showed that HPIValid performs poorly on the isolet dataset. A more thorough investigation of its structure may lead to a lower bound on the running time and, at the same time, point us to new ways to improve the current state of hitting set enumeration.

While also revealing faster algorithms, the main benefit of viewing data profiling from a hitting set perspective is that it provides techniques to make the *space* consumption of dependency discovery independent of the output size. Previously, this was a major issue when handling large datasets with many solutions. An enumeration routine that uses only linear space in the input was a key ingredient both in our algorithmic study in Chapter 4 as well as our engineering work in Chapter 7. This was the central improvement over previous work by Eiter and Gottlob [EG95] as well as Papenbrock and Naumann [PN17], respectively. In general, algorithms designed using the hitting set paradigm are better equipped for taking up the challenges in the age of big data.

The logical next step would be to bring the space reduction also to other parts of data profiling. This requires a careful review of the typical applications of multi-column dependencies and the existing processes and infrastructure.

If, for example, the discovered unique column combinations are handed down one-by-one in a profiling pipeline, then memory usage is not the main issue. One would rather minimize the delay of solutions in that case. However, there might be applications, especially in database query optimization, where the knowledge of all minimal UCCs is appreciated, but not all results are used at the same time. There, one may wish for the ability to look up, say, the UCCs containing a certain attribute while the current request is processed.

This is the prime use case of a data structure. Instead of producing all UCCs up front, a new kind of profiling algorithm could preprocess a structure that then provides fast access to the desired dependencies at query time. In data structure design, the trade-offs between preprocessing time, query time, and space are highly non-trivial, but also more relevant than for classical data profiling. Recently, the author took an interest in fault-tolerant data structures for graph problems that report the correct answer even if a few specified links in the underlying networked fail, see [Ahm+20; GV20; Hen+17; WY13]. We expect it to be very fruitful to transfer techniques from that area to enumeration. Conversely, ideas from enumeration may also be helpful to develop new data structures. At last, such a structure is but a succinct representation of all query results.

Bibliography

[AB09a] Kartik Anand and Ginestra Bianconi. **Entropy Measures for Networks: Toward an Information Theory of Complex Topologies**. *Physical Review E* 80 (2009). Article no. 045102. DOI: 10.1103/PhysRevE.80.045102 (see page 109).

[AB09b] Sanjeev Arora and Boaz Barak. **Computational Complexity: A Modern Approach**. New York City, NY, USA: Cambridge University Press, 2009. DOI: 10.1017/CBO9780511804090 (see page 9).

[AB96] Tatsuya Akutsu and Feng Bao. **Approximating Minimum Keys and Optimal Substructure Screens**. In: *Proceedings of the 2nd International Conference on Computing and Combinatorics (COCOON)*. 1996, 290–299. DOI: 10.1007/3-540-61332-3_163 (see pages 32, 37).

[Abe+18] Ziawasch Abedjan, Lukasz Golab, Felix Naumann, and Thorsten Papenbrock. **Data Profiling**. Vol. 10. Synthesis Lectures on Data Management. San Rafael, CA, USA: Morgan & Claypool Publishers, 2018. DOI: 10.2200/S00878ED1V01Y201810DTM052 (see pages 1, 6, 12, 13, 24, 26, 28, 82, 134, 135).

[Ahm+20] Reyan Ahmed, Greg Bodwin, Faryad Darabi Sahneh, Keaton Hamm, Mohammad Javad Latifi Jebelli, Stephen Kobourov, and Richard Spence. **Graph Spanners: A Tutorial Review**. *Computer Science Review* 37 (2020), Art. no. 100253. DOI: 10.1016/j.cosrev.2020.100253 (see page 165).

[AN11] Ziawasch Abedjan and Felix Naumann. **Advancing the Discovery of Unique Column Combinations**. In: *Proceedings of the 20th International Conference on Information and Knowledge Management (CIKM)*. 2011, 1565–1570. DOI: 10.1145/2063576.2063801 (see pages 13, 131, 134, 140).

[AQN14] Ziawasch Abedjan, Jorge-Arnulfo Quiané-Ruiz, and Felix Naumann. **Detecting Unique Column Combinations on Dynamic Data**. In: *Proceedings of the 30th International Conference on Data Engineering (ICDE)*. 2014, 1036–1047. DOI: 10.1109/ICDE.2014.6816721 (see pages 13, 131, 134).

[Ara+20] Naheed Anjum Arafat, Debabrota Basu, Laurent Decreusefond, and Stéphane Bressan. **Construction and Random Generation of Hypergraphs with Prescribed Degree and Dimension Sequences**. In: *Proceedings of the 31st International Conference on Database and Expert Systems Applications (DEXA)*. 2020, 130–145 (see page 110).

[Ara+21] Júlio Araújo, Marin Bougeret, Victor A. Campos, and Ignasi Sau. **Parameterized Complexity of Computing Maximum Minimal Blocking and Hitting Sets**. *CoRR* abs/2102.03404 (2021). ArXiv preprint. arXiv: 2102.03404 [cs.DS]. URL: https://arxiv.org/abs/2102.03404 (see page 59).

[Ash90] Robert B. Ash. **Information Theory**. Dover Books on Mathematics. Reprint of the Interscience Publishers 1965 edition. Mineola, NY, USA: Dover Publications, 1990 (see pages 92, 94).

[Bal+17] Marshall Ball, Alon Rosen, Manuel Sabin, and Prashant N. Vasudevan. **Average-Case Fine-Grained Hardness**. In: *Proceedings of the 49th Symposium on Theory of Computing, (STOC)*. 2017, 483–496. DOI: 10.1145/3055399.3055466 (see page 111).

[Baz+18] Cristina Bazgan, Ljiljana Brankovic, Katrin Casel, Henning Fernau, Klaus Jansen, Kim-Manuel Klein, Michael Lampis, Mathieu Liedloff, Jérôme Monnot, and Vangelis Th. Paschos. **The Many Facets of Upper Domination**. *Theoretical Computer Science* 717 (2018), 2–25. DOI: 10.1016/j.tcs.2017.05.042 (see pages 6, 56, 58, 59, 64).

[Baz+20] Cristina Bazgan, Ljiljana Brankovic, Katrin Casel, and Henning Fernau. **Domination Chain: Characterisation, Classical Complexity, Parameterised Complexity and Approximability**. *Discrete Applied Mathematics* 280 (2020), 23–42. DOI: 10.1016/j.dam.2019.10.005 (see page 64).

[BCH16] Michele Borassi, Pierluigi Crescenzi, and Michel Habib. **Into the Square: On the Complexity of Some Quadratic-time Solvable Problems**. *Electronic Notes in Theoretical Computer Science* 322 (2016), 51–67. DOI: 10.1016/j.entcs.2016.03.005 (see pages 108, 111, 153).

[BCK10] Michael Behrisch, Amin Coja-Oghlan, and Mihyun Kang. **The Order of the Giant Component of Random Hypergraphs**. *Random Structures and Algorithms* 36 (2010), 149–184. DOI: 10.1002/rsa.v36:2 (see page 110).

[BCK14] Michael Behrisch, Amin Coja-Oghlan, and Mihyun Kang. **Local Limit Theorems for the Giant Component of Random Hypergraphs**. *Combinatorics, Probability and Computing* 23 (2014), 331–366. DOI: 10.1017/S0963548314000017 (see page 110).

[BD75] Claude Berge and Pierre Duchet. **A Generalization of Gilmore's Theorem**. In: *Proceedings of the 2nd Czechoslovak Symposium on Recent Advances in Graph Theory*. 1975, 49–55 (see page 59).

[BDP15] Nicolas Boria, Federico Della Croce, and Vangelis Th. Paschos. **On the Max Min Vertex Cover Problem**. *Discrete Applied Mathematics* 196 (2015), 62–71. DOI: 10.1016/j.dam.2014.06.001 (see pages 58, 59).

[Bee+84] Catriel Beeri, Martin Dowd, Ronald Fagin, and Richard Statman. **On the Structure of Armstrong Relations for Functional Dependencies**. *Journal of the ACM* 31 (1984), 30–46. DOI: 10.1145/2422.322414 (see page 133).

[Ber+21] Julian Berger, Maximilian Böther, Vanja Doskoč, Jonathan Gadea Harder, Nicolas Klodt, Timo Kötzing, Winfried Lötzsch, Jannik Peters, Leon Schiller, Lars Seifert, Armin Wells, and Simon Wietheger. **Learning Languages with Decidable Hypotheses**. In: *Proceedings of the 17th Conference on Computability in Europe (CiE)*. 2021, 25–37. DOI: 10.1007/978-3-030-80049-9_3 (see page x).

[Ber89] Claude Berge. **Hypergraphs - Combinatorics of Finite Sets**. Vol. 45. North-Holland Mathematical Library. Amsterdam, Netherlands: North-Holland Publishing Company, 1989 (see pages 10, 59, 107).

[BF04] Aiden A. Bruen and Mario A. Forcinito. **Cryptography, Information Theory, and Error-Correction: A Handbook for the 21st Century**. New York City, NY, USA: Wiley-Interscience, 2004. DOI: 10.1002/9781118033296 (see page 110).

[BF99] Richard Beigel and Bin Fu. **Molecular Computing, Bounded Nondeterminism, and Efficient Recursion**. *Algorithmica* 25 (1999), 222–238. DOI: 10.1007/PL00008275 (see page 37).

[BFS16] Thomas Bläsius, Tobias Friedrich, and Martin Schirneck. **The Parameterized Complexity of Dependency Detection in Relational Databases**. In: *Proceedings of the 11th International Symposium on Parameterized and Exact Computation (IPEC)*. 2016, 6:1–6:13. DOI: 10.4230/LIPIcs.IPEC.2016.6 (see page viii).

[BFS20] Thomas Bläsius, Tobias Friedrich, and Martin Schirneck. **The Minimization of Random Hypergraphs**. In: *Proceedings of the 28th European Symposium on Algorithms (ESA)*. 2020, 21:1–21:15. DOI: 10.4230/LIPIcs.ESA.2020.21 (see page ix).

[BFS22] Thomas Bläsius, Tobias Friedrich, and Martin Schirneck. **The Complexity of Dependency Detection and Discovery in Relational Databases**. *Theoretical Computer Science* 900 (2022), 79–96 (see page viii).

[BGH98] Endre Boros, Vladimir Gurvich, and Peter L. Hammer. **Dual Subimplicants of Positive Boolean Functions**. *Optimization Methods and Software* 10 (1998), 147–156. DOI: 10.1080/10556789808805708 (see pages 37, 60, 64, 65).

[BI95] Jan C. Bioch and Toshihide Ibaraki. **Complexity of Identification and Dualization of Positive Boolean Functions**. *Information and Computation* 123 (1995), 50–63. DOI: 10.1006/inco.1995.1157 (see pages 3, 16, 37, 152).

[Bia07] Ginestra Bianconi. **The Entropy of Randomized Network Ensembles**. *Europhysics Letters* 81 (2007), Art. no. 28005. DOI: 10.1209/0295-5075/81/28005 (see page 110).

[Bil+21a] Davide Bilò, Sarel Cohen, Tobias Friedrich, and Martin Schirneck. **Near-Optimal Deterministic Single-Source Distance Sensitivity Oracles**. In: *Proceedings of the 29th European Symposium on Algorithms (ESA)*. 2021, 18:1–18:17. DOI: 10.4230/LIPIcs.ESA.2021.18 (see page x).

[Bil+21b] Davide Bilò, Sarel Cohen, Tobias Friedrich, and Martin Schirneck. **Space-Efficient Fault-Tolerant Diameter Oracles**. In: *Proceedings of the 46th International Symposium on Mathematical Foundations of Computer Science (MFCS)*. 2021, 18:1–18:16. DOI: 10.4230/LIPIcs.MFCS.2021.18 (see page x).

[Bil+22a] Davide Bilò, Katrin Casel, Keerti Choudhary, Sarel Cohen, Tobias Friedrich, J.A. Gregor Lagodzinski, Martin Schirneck, and Simon Wietheger. **Fixed-Parameter Sensitivity Oracles**. In: *Proceedings of the 13th Innovations in Theoretical Computer Science Conference (ITCS)*. 2022, 23:1–23:18. DOI: 10.4230/LIPIcs.ITCS.2022.23 (see page x).

[Bil+22b] Davide Bilò, Keerti Choudhary, Sarel Cohen, Tobias Friedrich, and Martin Schirneck. **Deterministic Sensitivity Oracles for Diameter, Eccentricities and All Pairs Distances**. In: *Proceedings of the 49th International Colloquium on Automata, Languages, and Programming (ICALP)*. 2022, 22:1–22:19. DOI: 10.4230/LIPIcs.ICALP.2022.22 (see page x).

[Bir+20] Johann Birnick, Thomas Bläsius, Tobias Friedrich, Felix Naumann, Thorsten Papenbrock, and Martin Schirneck. **Hitting Set Enumeration with Partial Information for Unique Column Combination Discovery**. *Proceedings of the VLDB Endowment* 13 (2020), 2270–2283. DOI: 10.14778/3407790.3407824 (see page ix).

[Bjö+15] Andreas Björklund, Petteri Kaski, Łukasz Kowalik, and Juho Lauri. **Engineering Motif Search for Large Graphs**. In: *Proceedings of the 17th Meeting on Algorithm Engineering and Experiments (ALENEX)*. 2015, 104–118. DOI: 10.1137/1.9781611973754.10 (see page 60).

[BKN17] Tobias Bleifuß, Sebastian Kruse, and Felix Naumann. **Efficient Denial Constraint Discovery with Hydra**. *Proceedings of the VLDB Endowment* 11 (2017), 311–323. DOI: 10.14778/3157794.3157800 (see pages 153, 161, 164).

[Blä+19a] Thomas Bläsius, Philipp Fischbeck, Tobias Friedrich, and Martin Schirneck. **Understanding the Effectiveness of Data Reduction in Public Transportation Networks**. In: *Proceedings of the 16th Workshop on Algorithms and Models for the Web Graph (WAW)*. 2019, 87–101. DOI: 10.1007/978-3-030-25070-6_7 (see page x).

[Blä+19b] Thomas Bläsius, Tobias Friedrich, Julius Lischeid, Kitty Meeks, and Martin Schirneck. **Efficiently Enumerating Hitting Sets of Hypergraphs Arising in Data Profiling**. In: *Proceedings of the 21st Meeting on Algorithm Engineering and Experiments (ALENEX)*. 2019, 130–143. DOI: 10.1137/1.9781611975499.11 (see page ix).

[Blä+22] Thomas Bläsius, Tobias Friedrich, Julius Lischeid, Kitty Meeks, and Martin Schirneck. **Efficiently Enumerating Hitting Sets of Hypergraphs Arising in Data Profiling**. *Journal of Computer and System Sciences* 124 (2022), 192–213. DOI: 10.1016/j.jcss.2021.10.002 (see page ix).

[BLS15] Golnaz Badkobeh, Per Kristian Lehre, and Dirk Sudholt. **Black-Box Complexity of Parallel Search with Distributed Populations**. In: *Proceedings of the 2015 Conference on Foundations of Genetic Algorithms (FOGA)*. 2015, 3–15. DOI: 10.1145/2725494.2725504 (see page 22).

[Bol01] Béla Bollobás. **Random Graphs**. 2nd Edition. Studies in Advanced Mathematics. Cambridge, UK: Cambridge University Press, 2001. DOI: 10.1017/CBO9780511814068 (see pages 107, 109).

[Bol80] Béla Bollobás. **A Probabilistic Proof of an Asymptotic Formula for the Number of Labelled Regular Graphs**. *European Journal of Combinatorics* 1 (1980), 311–316. DOI: 10.1016/S0195-6698(80)80030-8 (see page 110).

[Bor+00] Endre Boros, Khaled M. Elbassioni, Vladimir Gurvich, and Leonid G. Khachiyan. **An Efficient Incremental Algorithm for Generating All Maximal Independent Sets in Hypergraphs of Bounded Dimension**. *Parallel Processing Letters* 10 (2000), 253–266. DOI: 10.1142/S0129626400000251 (see page 55).

[Bor+02] Endre Boros, Khaled M. Elbassioni, Vladimir Gurvich, Leonid G. Khachiyan, and Kazuhisa Makino. **Dual-Bounded Generating Problems: All Minimal Integer Solutions for a Monotone System of Linear Inequalities**. *SIAM Journal on Computing* 31 (2002), 1624–1643. DOI: 10.1137/S0097539701388768 (see page 26).

[Bri19] Karl Bringmann. **Fine-Grained Complexity Theory (Tutorial)**. In: *Proceedings of the 36th Symposium on Theoretical Aspects of Computer Science (STACS)*. 2019, 4:1–4:7. DOI: 10.4230/LIPIcs.STACS.2019.4 (see page 71).

[Car+16] Marco L. Carmosino, Jiawei Gao, Russell Impagliazzo, Ivan Mihajlin, Ramamohan Paturi, and Stefan Schneider. **Nondeterministic Extensions of the Strong Exponential Time Hypothesis and Consequences for Non-reducibility**. In: *Proceedings of the 7th Conference on Innovations in Theoretical Computer Science (ITCS)*. 2016, 261–270. DOI: 10.1145/2840728.2840746 (see pages 8, 56, 57, 75, 76).

[Cas+18] Katrin Casel, Henning Fernau, Mehdi Khosravian Ghadikolaei, Jérôme Monnot, and Florian Sikora. **On the Complexity of Solution Extension of Optimization Problems**. *CoRR* abs/1810.04553 (2018). ArXiv preprint. arXiv: 1810.04553 [cs.CC]. URL: http://arxiv.org/abs/1810.04553 (see page 56).

[Cas+19] Katrin Casel, Henning Fernau, Mehdi Khosravian Ghadikolaei, Jérôme Monnot, and Florian Sikora. **Extension of Vertex Cover and Independent Set in Some Classes of Graphs and Generalizations**. In: *Proceedings of the 11th International Conference on Algorithms and Complexity (CIAC)*. 2019, 124–136. DOI: 10.1007/978-3-030-17402-6_11 (see page 64).

[Cas+21] Katrin Casel, Henning Fernau, Mehdi Khosravian Ghadikolaei, Jérôme Monnot, and Florian Sikora. **On the Complexity of Solution Extension of Optimization Problems**. *Theoretical Computer Science* (2021). In press. DOI: 10.1016/j.tcs.2021.10.017 (see pages 5, 56).

[CFP19] Colin Cooper, Alan M. Frieze, and Wesley Pegden. **On the Rank of a Random Binary Matrix**. *Electronic Journal of Combinatorics* 26 (2019). Article no. P4.12. DOI: 10.37236/8092 (see page 110).

[CFP84] Marco A. Casanova, Ronald Fagin, and Christos H. Papadimitriou. **Inclusion Dependencies and Their Interaction with Functional Dependencies**. *Journal of Computer and System Sciences* 28 (1984), 29–59. DOI: 10.1016/0022-0000(84)90075-8 (see page 24).

[CH97] Nadia Creignou and Jean-Jacques Hébrard. **On Generating All Solutions of Generalized Satisfiability Problems**. *RAIRO - Theoretical Informatics and Applications* 31 (1997), 499–511. DOI: 10.1051/ita/1997310604991 (see page 60).

[Che+06] Jianer Chen, Xiuzhen Huang, Iyad A. Kanj, and Ge Xia. **Strong Computational Lower Bounds via Parameterized Complexity**. *Journal of Computer and System Sciences* 72 (2006), 1346–1367. DOI: 10.1016/j.jcss.2006.04.007 (see pages 19, 59, 72, 73).

[Che+90] Grant A. Cheston, Gerd H. Fricke, Stephen T. Hedetniemi, and David Pokrass Jacobs. **On the Computational Complexity of Upper Fractional Domination**. *Discrete Applied Mathematics* 27 (1990), 195–207. DOI: 10.1016/0166-218X(90)90065-K (see page 58).

[Cho20] Philip S. Chodrow. **Configuration Models of Random Hypergraphs**. *Journal of Complex Networks* 8 (2020). Article no. cnaa018. DOI: 10.1093/comnet/cnaa018 (see page 110).

[CKS86] Stavros S. Cosmadakis, Paris C. Kanellakis, and Nicolas Spyratos. **Partition Semantics for Relations**. *Journal of Computer and System Sciences* 33 (1986), 203–233. DOI: 10.1016/0022-0000(86)90019-X (see page 140).

[CM03] Carlos Cotta and Pablo Moscato. **The k-Feature Set Problem is $W[2]$-complete**. *Journal of Computer and System Sciences* 67 (2003), 686–690. DOI: 10.1016/S0022-0000(03)00081-3 (see pages 32, 34, 37).

[CN08] Pierre Colomb and Lhouari Nourine. **About Keys of Formal Context and Conformal Hypergraph**. In: *Proceedings of the 6th International Conference on Formal Concept Analysis (ICFCA)*. 2008, 140–149. DOI: 10.1007/978-3-540-78137-0_10 (see page 58).

[Con+20] Alessio Conte, Roberto Grossi, Andrea Marino, and Luca Versari. **Sublinear-Space and Bounded-Delay Algorithms for Maximal Clique Enumeration in Graphs**. *Algorithmica* 82 (2020), 1547–1573. DOI: 10.1007/s00453-019-00656-8 (see page 38).

[Cra38] Harald Cramér. **Sur un nouveau théorème-limite de la théorie des probabilités**. In: *Actualités scientifiques et industrielles*. Vol. 763. Colloque consacré à la théorie des probabilités. (On a New Limit Theorem in Probability.) In French. 1938, 5–23 (see pages 92, 105).

[Cre+17] Nadia Creignou, Arne Meier, Julian-Steffen Müller, Johannes Schmidt, and Heribert Vollmer. **Paradigms for Parameterized Enumeration**. *Theory of Computing Systems* 60 (2017), 737–758. DOI: 10.1007/s00224-016-9702-4 (see page 4).

[Cre+19] Nadia Creignou, Markus Kröll, Reinhard Pichler, Sebastian Skritek, and Heribert Vollmer. **A Complexity Theory for Hard Enumeration Problems**. *Discrete Applied Mathematics* 268 (2019), 191–209. DOI: 10.1016/j.dam.2019.02.025 (see page 37).

[CS19] Florent Capelli and Yann Strozecki. **Incremental Delay Enumeration: Space and Time**. *Discrete Applied Mathematics* 268 (2019), 179–190. DOI: 10.1016/j.dam.2018.06.038 (see pages 15, 16, 38).

[CT06] Thomas M. Cover and Joy A. Thomas. **Elements of Information Theory**. 2nd Edition. Wiley Series in Telecommunications and Signal Processing. New York City, NY, USA: Wiley-Interscience, 2006. DOI: 10.1002/047174882X (see pages 20, 109, 111).

[Cyg+15] Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. **Parameterized Algorithms**. Cham, Switzerland: Springer, 2015. DOI: 10.1007/978-3-319-21275-3 (see pages 16, 24, 73).

[CZ06] Jianer Chen and Fenghui Zhang. **On Product Covering in 3-Tier Supply Chain Models: Natural Complete Problems for $W[3]$ and $W[4]$**. *Theoretical Computer Science* 363 (2006), 278–288. DOI: 10.1016/j.tcs.2006.07.016 (see page 5).

[Dam06] Peter Damaschke. **Parameterized Enumeration, Transversals, and Imperfect Phylogeny Reconstruction**. *Theoretical Computer Science* 351 (2006), 337–350. DOI: 10.1016/j.tcs.2005.10.004 (see page 4).

[Dam11] Peter Damaschke. **Parameterized Algorithms for Double Hypergraph Dualization with Rank Limitation and Maximum Minimal Vertex Cover**. *Discrete Optimization* 8 (2011), 18–24. DOI: 10.1016/j.disopt.2010.02.006 (see pages 58, 59).

[Dat03] Christopher J. Date. **An Introduction to Database Systems**. 8th Edition. Boston, MA, USA: Addison-Wesley Longman Publishing, 2003 (see pages 13, 28).

[Dem+98] János Demetrovics, Gyula O. H. Katona, Dezso Miklós, Oleg Seleznjev, and Bernhard Thalheim. **Asymptotic Properties of Keys and Functional Dependencies in Random Databases**. *Theoretical Computer Science* 190 (1998), 151–166. DOI: 10.1016/S0304-3975(97)00089-3 (see page 111).

[DF13] Rodney G. Downey and Michael R. Fellows. **Fundamentals of Parameterized Complexity**. Texts in Computer Science. London, UK: Springer, 2013. DOI: 10.1007/978-1-4471-5559-1 (see pages 3, 16, 24, 29, 49).

[DF95a] Rodney G. Downey and Michael R. Fellows. **Fixed-Parameter Tractability and Completeness I: Basic Results**. *SIAM Journal on Computing* 24 (1995), 873–921. DOI: 10.1137/S0097539792228228 (see pages 4, 44).

[DF95b] Rodney G. Downey and Michael R. Fellows. **Fixed-Parameter Tractability and Completeness II: On Completeness for $W[1]$**. *Theoretical Computer Science* 141 (1995), 109–131. DOI: 10.1016/0304-3975(94)00097-3 (see page 44).

[DF98] Rodney G. Downey and Michael R. Fellows. **Threshold Dominating Sets and an Improved Characterization of $W[2]$**. *Theoretical Computer Science* 209 (1998), 123–140. DOI: 10.1016/S0304-3975(97)00101-1 (see page 34).

[DF99] Rodney G. Downey and Michael R. Fellows. **Parameterized Complexity**. Monographs in Computer Science. New York City, NY, USA: Springer, 1999. DOI: 10.1007/978-1-4612-0515-9 (see page 3).

[DK20] Vanja Doskoč and Timo Kötzing. **Cautious Limit Learning**. In: *Proceedings of the 31st International Conference on Algorithmic Learning Theory (ALT)*. 2020, 251–276. URL: http://proceedings.mlr.press/v117/doskoc20a (see page x).

[DK21a] Vanja Doskoč and Timo Kötzing. **Mapping Monotonic Restrictions in Inductive Inference**. In: *Proceedings of the 17th Conference on Computability in Europe (CiE)*. 2021, 146–157. DOI: 10.1007/978-3-030-80049-9_13 (see page x).

[DK21b] Vanja Doskoč and Timo Kötzing. **Normal Forms for Semantically Witness-Based Learners in Inductive Inference**. In: *Proceedings of the 17th Conference on Computability in Europe (CiE)*. 2021, 158–168. DOI: 10.1007/978-3-030-80049-9_14 (see page x).

[DLP21] Louis Dublois, Michael Lampis, and Vangelis Th. Paschos. **Upper Dominating Set: Tight Algorithms for Pathwidth and Sub-exponential Approximation**. In: *Proceedings of the 12th International Conference on Algorithms and Complexity (CIAC)*. 2021, 202–215. DOI: 10.1007/978-3-030-75242-2_14 (see page 59).

[DLV20] Mina Dalirrooyfard, Andrea Lincoln, and Virginia Vassilevska Williams. **New Techniques for Proving Fine-Grained Average-Case Hardness**. In: *Proceedings of the 61st Symposium on Foundations of Computer Science (FOCS)*. 2020, 774–785. DOI: 10.1109/FOCS46700.2020.00077 (see page 111).

[DMP99] Carlos Domingo, Nina Mishra, and Leonard Pitt. **Efficient Read-Restricted Monotone CNF/DNF Dualization by Learning with Membership Queries**. *Machine Learning* 37 (1999), 89–110. DOI: 10.1023/A:1007627028578 (see pages 26, 37).

[Doe+17] Benjamin Doerr, Philipp Fischbeck, Clemens Frahnow, Tobias Friedrich, Timo Kötzing, and Martin Schirneck. **Island Models Meet Rumor Spreading**. In: *Proceedings of the 2017 Genetic and Evolutionary Computation Conference (GECCO)*. 2017, 1359–1366. DOI: 10.1145/3071178.3071206 (see page x).

[Doe+19] Benjamin Doerr, Philipp Fischbeck, Clemens Frahnow, Tobias Friedrich, Timo Kötzing, and Martin Schirneck. **Island Models Meet Rumor Spreading**. *Algorithmica* 81 (2019), 886–915. ISSN: 1432-0541. DOI: 10.1007/s00453-018-0445-2 (see page x).

[Doe20] Benjamin Doerr. **Probabilistic Tools for the Analysis of Randomized Optimization Heuristics**, 1–87. In: *Theory of Evolutionary Computation*. Ed. by Benjamin Doerr and Frank Neumann. Natural Computing Series. Basel, Switzerland: Springer International, 2020. DOI: 10.1007/978-3-030-29414-4_1 (see pages 92, 103).

[DP09] Devdatt Dubhashi and Alessandro Panconesi. **Concentration of Measure for the Analysis of Randomized Algorithms**. New York City, NY, USA: Cambridge University Press, 2009 (see pages 92, 103).

[DR94] Scott Davies and Stuart Russell. **NP-Completeness of Searches for Smallest Possible Feature Sets**. In: *AAAI Symposium on Intelligent Relevance*. Technical Report. 1994. URL: https://www.aaai.org/Papers/Symposia/Fall/1994/FS-94-02/FS94-02-011.pdf (see pages 28, 31, 32).

[DS14] Irit Dinur and David Steurer. **Analytical Approach to Parallel Repetition**. In: *Proceedings of the 46th Symposium on Theory of Computing (STOC)*. 2014, 624–633. DOI: 10.1145/2591796.2591884 (see pages 29, 36).

[DS89] Jean-Dominique Deuschel and Daniel W. Stroock. **Large Deviations**. Vol. 137. Pure and Applied Mathematics. Boston, MA, USA: Academic Press, 1989. DOI: 10.1016/S0079-8169(08)61645-1 (see pages 92, 105).

[DT87] János Demetrovics and Vu Duc Thi. **Keys, Antikeys and Prime Attributes**. *Annales Universitatis Scientiarum Budapestinensis de Rolando Eötvös Nominatae Sectio Computatorica* 8 (1987), 35–52 (see pages 2, 26, 132, 135).

[DZ10] Amir Dembo and Ofer Zeitouni. **Large Deviations Techniques and Applications**. 2nd Edition. Vol. 38. Stochastic Modelling and Applied Probability. Berlin & Heidelberg, Germany: Springer, 2010. DOI: 10.1007/978-3-642-03311-7 (see pages 92, 105).

[EG95] Thomas Eiter and Georg Gottlob. **Identifying the Minimal Transversals of a Hypergraph and Related Problems**. *SIAM Journal on Computing* 24 (1995), 1278–1304. DOI: 10.1137/S0097539793250299 (see pages 2, 3, 26, 31, 37, 41, 42, 55, 58, 134, 164).

[EGM03] Thomas Eiter, Georg Gottlob, and Kazuhisa Makino. **New Results on Monotone Dualization and Generating Hypergraph Transversals**. *SIAM Journal on Computing* 32 (2003), 514–537. DOI: 10.1137/S009753970240639X (see pages 11, 37, 55, 63).

[EHR08] Khaled M. Elbassioni, Matthias Hagen, and Imran Rauf. **Some Fixed-Parameter Tractable Classes of Hypergraph Duality and Related Problems**. In: *Proceedings of the 3rd International Workshop on Parameterized and Exact Computation (IWPEC)*. 2008, 91–102. DOI: 10.1007/978-3-540-79723-4_10 (see pages 37, 38, 63).

[EMG08] Thomas Eiter, Kazuhisa Makino, and Georg Gottlob. **Computational Aspects of Monotone Dualization: A Brief Survey**. *Discrete Applied Mathematics* 156 (2008), 2035–2049. DOI: 10.1016/j.dam.2007.04.017 (see pages 2, 26, 134, 135, 152).

[ER59] Paul Erdős and Alfréd Rényi. **On Random Graphs I**. *Publicationes Mathematicae Debrecen* 6 (1959), 290–297 (see page 109).

[Fel+09] Michael R. Fellows, Danny Hermelin, Frances A. Rosamond, and Stéphane Vialette. **On the Parameterized Complexity of Multiple-Interval Graph Problems**. *Theoretical Computer Science* 410 (2009), 53–61. DOI: 10.1016/j.tcs.2008.09.065 (see page 66).

[Fer02] Henning Fernau. **On Parameterized Enumeration**. In: *Proceedings of the 8th International Computing and Combinatorics Conference (COCOON)*. 2002, 564–573. DOI: 10.1007/3-540-45655-4_60 (see page 4).

[Fer05] Henning Fernau. **Parameterized Algorithmics: A Graph-Theoretic Approach**. University of Tübingen, Habilitationsschrift. 2005. URL: https://informatik.uni-trier.de/~fernau/papers/habil.pdf (see page 58).

[FF75] Raymond Fadous and John Forsyth. **Finding Candidate Keys for Relational Data Bases**. In: *Proceedings of the 1975 SIGMOD International Conference on Management of Data (SIGMOD)*. 1975, 203–210 (see page 133).

[FG06] Jörg Flum and Martin Grohe. **Parameterized Complexity Theory**. Texts in Theoretical Computer Science. An EATCS Series. Berlin & Heidelberg, Germany: Springer, 2006. DOI: 10.1007/3-540-29953-X (see pages 16, 76).

[Fil83] James A. Fill. **Convergence Rates Related to the Strong Law of Large Numbers**. *Annals of Probability* 11 (1983), 123–142. DOI: 10.1214/aop/1176993663 (see page 105).

[FK96] Michael L. Fredman and Leonid G. Khachiyan. **On the Complexity of Dualization of Monotone Disjunctive Normal Forms**. *Journal of Algorithms* 21 (1996), 618–628. DOI: 10.1006/jagm.1996.0062 (see pages 3, 29, 37).

[FM13] Alan M. Frieze and Dhruv Mubayi. **Coloring Simple Hypergraphs**. *Journal of Combinatorial Theory, Series B* 103 (2013), 767–794. DOI: 10.1016/j.jctb.2013.09.003 (see page 10).

[Fom+08] Fedor V. Fomin, Fabrizio Grandoni, Artem V. Pyatkin, and Alexey A. Stepanov. **Combinatorial Bounds via Measure and Conquer: Bounding Minimal Dominating Sets and Applications**. *ACM Transactions on Algorithms* 5 (2008), 9:1–9:17. DOI: 10.1145/1435375.1435384 (see page 135).

[Fri+16] Tobias Friedrich, Timo Kötzing, Martin S. Krejca, Samadhi Nallaperuma, Frank Neumann, and Martin Schirneck. **Fast Building Block Assembly by Majority Vote Crossover**. In: *Proceedings of the 2016 Genetic and Evolutionary Computation Conference (GECCO)*. 2016, 661–668. DOI: 10.1145/2908812.2908884 (see page x).

[Fri+17] Tobias Friedrich, Timo Kötzing, J.A. Gregor Lagodzinski, Frank Neumann, and Martin Schirneck. **Analysis of the (1+1) EA on Subclasses of Linear Functions under Uniform and Linear Constraints**. In: *Proceedings of the 14th Conference on Foundations of Genetic Algorithms (FOGA)*. 2017, 45–54. DOI: 10.1145/3040718.3040728 (see page x).

[Fri+20] Tobias Friedrich, Timo Kötzing, J.A. Gregor Lagodzinski, Frank Neumann, and Martin Schirneck. **Analysis of the (1+1) EA on Subclasses of Linear Functions under Uniform and Linear Constraints**. *Theoretical Computer Science* 832 (2020), 3–19. DOI: 10.1016/j.tcs.2018.04.051 (see page x).

[Fri+22] Tobias Friedrich, Timo Kötzing, Aishwarya Radhakrishnan, Leon Schiller, Martin Schirneck, Georg Tennigkeit, and Simon Wietheger. **Crossover for Cardinality Constrained Optimization**. In: *Proceedings of the 2022 Genetic and Evolutionary Computation Conference (GECCO)*. 2022, 1399–1407. DOI: 10.1145/3512290.3528713 (see page x).

[Fro+16] Vincent Froese, René van Bevern, Rolf Niedermeier, and Manuel Sorge. **Exploiting Hidden Structure in Selecting Dimensions That Distinguish Vectors**. *Journal of Computer and System Sciences* 82 (2016), 521–535. DOI: 10.1016/j.jcss.2015.11.011 (see page 32).

[Gao+18] Jiawei Gao, Russell Impagliazzo, Antonina Kolokolova, and R. Ryan Williams. **Completeness for First-Order Properties on Sparse Structures With Algorithmic Applications**. *ACM Transactions on Algorithms* 15 (2018), 23:1–23:35. DOI: 10.1145/3196275 (see pages 19, 111, 153).

[GB06] Yves Grandvalet and Yoshua Bengio. **Entropy Regularization**, 151–168. In: *Semi-Supervised Learning*. Ed. by Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien. Cambridge, MA, USA: MIT Press, 2006. DOI: 10.7551/mitpress/9780262033589.001.0001 (see page 110).

[GB85] Héctor Garcia-Molina and Daniel Barbara. **How to Assign Votes in a Distributed System**. *Journal of the ACM* 32 (1985), 841–860. DOI: 10.1145/4221.4223 (see page 26).

[Gia+02] Chris Giannella, Mehmet M. Dalkilic, Dennis P. Groth, and Edward L. Robertson. **Improving Query Evaluation with Approximate Functional Dependency Based Decompositions**. In: *Proceedings of the 19th British National Conference on Databases (BNCOD)*. Vol. 2405. 2002, 26–41. DOI: 10.1007/3-540-45495-0_3 (see page 24).

[Gil59] Edgar N. Gilbert. **Random Graphs**. *Annals of Mathematical Statistics* 30 (1959), 1141–1144. DOI: doi:10.1214/aoms/1177706098 (see page 109).

[GL08] Diego Garlaschelli and Maria I. Loffredo. **Maximum Likelihood: Extracting Unbiased Information from Complex Networks**. *Physical Review E* 78 (2008). Article no. 015101. DOI: 10.1103/PhysRevE.78.015101 (see page 110).

[Gol+22] Petr A. Golovach, Christian Komusiewicz, Dieter Kratsch, and Van Bang Le. **Refined Notions of Parameterized Enumeration Kernels with Applications to Matching Cut Enumeration**. *Journal of Computer and System Sciences* 123 (2022), 76–102. DOI: 10.1016/j.jcss.2021.07.005 (see page 4).

[GUW08] Héctor Garcia-Molina, Jeffrey D. Ullman, and Jennifer Widom. **Database Systems: The Complete Book**. 2nd Edition. Upper Saddle River, NJ, USA: Prentice Hall Press, 2008 (see page 13).

[GV17] Andrew Gainer-Dewar and Paola Vera-Licona. **The Minimal Hitting Set Generation Problem: Algorithms and Computation**. *SIAM Journal on Discrete Mathematics* 31 (2017), 63–100. DOI: 10.1137/15M1055024 (see pages 2, 3, 26, 135, 139, 151, 153).

[GV20] Fabrizio Grandoni and Virginia Vassilevska Williams. **Faster Replacement Paths and Distance Sensitivity Oracles**. *ACM Transaction on Algorithms* 16 (2020), 15:1–15:25. DOI: 10.1145/3365835 (see page 165).

[Har01] Peter Harremoës. **Binomial and Poisson Distributions as Maximum Entropy Distributions**. *IEEE Transactions on Information Theory* 47 (2001), 2039–2041. DOI: 10.1109/18.930936 (see page 112).

[Hei+13] Arvid Heise, Jorge-Arnulfo Quiané-Ruiz, Ziawasch Abedjan, Anja Jentzsch, and Felix Naumann. **Scalable Discovery of Unique Column Combinations**. *Proceedings of the VLDB Endowment* 7 (2013), 301–312. DOI: 10.14778/2732240.2732248 (see pages 13, 90, 131, 134, 140).

[Hen+17] Monika Henzinger, Andrea Lincoln, Stefan Neumann, and Virginia Vassilevska Williams. **Conditional Hardness for Sensitivity Problems**. In: *Proceedings of the 8th Innovations in Theoretical Computer Science Conference (ITCS)*. 2017, 26:1–26:31. DOI: 10.4230/LIPIcs.ITCS.2017.26 (see page 165).

[Hoe63] Wassily Hoeffding. **Probability Inequalities for Sums of Bounded Random Variables**. *Journal of the American Statistical Association* 58 (1963), 13–30 (see pages 7, 92).

[Hof16] Remco van der Hofstad. **Random Graphs and Complex Networks**. Vol. 1. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge, UK: Cambridge University Press, 2016. DOI: 10.1017/9781316779422 (see page 107).

[HSL21] Miika Hannula, Bor-Kuan Song, and Sebastian Link. **An Algorithm for the Discovery of Independence from Data**. *CoRR* abs/2101.02502 (2021). ArXiv preprint. arXiv: 2101.02502 [cs.DB]. URL: https://arxiv.org/abs/2101.02502 (see pages 5, 26).

[Huh+99] Ykä Huhtala, Juha Kärkkäinen, Pasi Porkka, and Hannu Toivonen. **TANE: An Efficient Algorithm for Discovering Functional and Approximate Dependencies**. *The Computer Journal* 42 (1999), 100–111. DOI: 10.1093/comjnl/42.2.100 (see pages 135, 140, 143).

[Ily+04] Ihab F. Ilyas, Volker Markl, Peter J. Haas, Paul Brown, and Ashraf Aboulnaga. **CORDS: Automatic Discovery of Correlations and Soft Functional Dependencies**. In: *Proceedings of the 2004 International Conference on Management of Data (SIGMOD)*. 2004, 647–658. DOI: 10.1145/1007568.1007641 (see page 24).

[IP01] Russell Impagliazzo and Ramamohan Paturi. **On the Complexity of *k*-SAT**. *Journal of Computer and System Sciences* 62 (2001), 367–375. DOI: 10.1006/jcss.2000.1727 (see pages 5, 18, 19, 73).

[IPZ01] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. **Which Problems Have Strongly Exponential Complexity?** *Journal of Computer and System Sciences* 63 (2001), 512–530. DOI: 10.1006/jcss.2001.1774 (see pages 5, 18, 73).

[Jay57a] Edwin T. Jaynes. **Information Theory and Statistical Mechanics**. *Physical Review Series II* 106 (1957), 620–630. DOI: 10.1103/PhysRev.106.620 (see page 110).

[Jay57b] Edwin T. Jaynes. **Information Theory and Statistical Mechanics II**. *Phyical Review Series II* 108 (1957), 171–190. DOI: 10.1103/PhysRev.108.171 (see page 110).

[JPY88] David S. Johnson, Christos H. Papadimitriou, and Mihalis Yannakakis. **On Generating All Maximal Independent Sets**. *Information Processing Letters* 27 (1988), 119–123. DOI: 10.1016/0020-0190(88)90065-8 (see pages 11, 63, 82).

[Kan+14] Mamadou M. Kanté, Vincent Limouzy, Arnaud Mary, and Lhouari Nourine. **On the Enumeration of Minimal Dominating Sets and Related Notions**. *SIAM Journal on Discrete Mathematics* 28 (2014), 1916–1929. DOI: 10.1137/120862612 (see page 64).

[Kan+92] Martti Kantola, Heikki Mannila, Kari-Jouko Räihä, and Harri Siirtola. **Discovering Functional and Inclusion Dependencies in Relational Databases**. *International Journal of Intelligent Systems* 7 (1992), 591–607. DOI: 10.1002/int.4550070703 (see pages 28, 45).

[Kar72] Richard M. Karp. **Reducibility Among Combinatorial Problems**. In: *Proceedings of a Symposium on the Complexity of Computer Computations*. 1972, 85–103. DOI: 10.1007/978-1-4684-2001-2_9 (see page 29).

[Kat12] Gyula O. H. Katona. **Random Databases with Correlated Data**, 29–35. In: *Conceptual Modelling and Its Theoretical Foundations: Essays Dedicated to Bernhard Thalheim on the Occasion of His 60th Birthday*. Ed. by Antje Düsterhöft, Meike Klettke, and Klaus-Dieter Schewe. Festschrift. Berlin & Heidelberg, Germany: Springer, 2012. DOI: 10.1007/978-3-642-28279-9_4 (see page 111).

[Kat13] Gyula O. H. Katona. **Testing Functional Connection Between Two Random Variables**, 335–348. In: *Prokhorov and Contemporary Probability Theory*. Ed. by Albert N. Shiryaev, S. R. Srinivasa Varadhan, and Ernst L. Presman. Festschrift. Berlin & Heidelberg, Germany: Springer, 2013. DOI: 10.1007/978-3-642-33549-5_20 (see page 111).

[Kes09] Hiremagalur K. Kesavan. **Jaynes' Maximum Entropy Principle**, 1779–1782. In: *Encyclopedia of Optimization*. Ed. by Christodoulos A. Floudas and Panos M. Pardalos. Boston, MA, USA: Springer, 2009. DOI: 10.1007/978-0-387-74759-0_312 (see page 110).

[Kha+07] Leonid Khachiyan, Endre Boros, Khaled M. Elbassioni, and Vladimir Gurvich. **A Global Parallel Algorithm for the Hypergraph Transversal Problem**. *Information Processing Letters* 101 (2007), 148–155. DOI: 10.1016/j.ipl.2006.09.006 (see page 55).

[KKS21] Ardalan Khazraei, Timo Kötzing, and Karen Seidel. **Towards a Map for Incremental Learning in the Limit from Positive and Negative Information**. In: *Proceedings of the 17th Conference on Computability in Europe (CiE)*. 2021, 273–284. DOI: 10.1007/978-3-030-80049-9_25 (see page x).

[KŁ02] Michał Karoński and Tomasz Łuczak. **The Phase Transition in a Random Hypergraph**. *Journal of Computational and Applied Mathematics* 142 (2002), 125–135. DOI: 10.1016/S0377-0427(01)00464-2 (see page 110).

[Kla00] Bernhard Klar. **Bounds on Tail Probabilities of Discrete Distributions**. *Probability in the Engineering and Informational Sciences* 14 (2000), 161–171 (see pages 93, 94).

[KLZ16] Henning Köhler, Sebastian Link, and Xiaofang Zhou. **Discovering Meaningful Certain Keys from Incomplete and Inconsistent Relations**. *Data Engineering Bulletin* 39 (2016), 21–37 (see pages 134, 135).

[Köh+16] Henning Köhler, Uwe Leck, Sebastian Link, and Xiaofang Zhou. **Possible and Certain Keys for SQL**. *The VLDB Journal* 25 (2016), 571–596. DOI: 10.1007/s00778-016-0430-9 (see pages 13, 38, 82).

[KP16] Timo Kötzing and Raphaela Palenta. **A Map of Update Constraints in Inductive Inference**. *Theoretical Computer Science* 650 (2016), 4–24. DOI: 10.1016/j.tcs.2016.07.028 (see page x).

[KPN21] Jan Kossmann, Thorsten Papenbrock, and Felix Naumann. **Data Dependencies for Query Optimization: A Survey**. *The VLDB Journal* (2021). In press. DOI: 10.1007/s00778-021-00676-3 (see pages 1, 24).

[Kru+16] Sebastian Kruse, Thorsten Papenbrock, Hazar Harmouch, and Felix Naumann. **Data Anamnesis: Admitting Raw Data into an Organization**. *IEEE Data Engineering Bulletin* 39 (2016), 8–20. URL: http://sites.computer.org/debull/A16june/p8.pdf (see page 55).

[KS03] Dimitris J. Kavvadias and Elias C. Stavropoulos. **Monotone Boolean Dualization is in co-NP[$\log^2 n$]**. *Information Processing Letters* 85 (2003), 1–6. DOI: 10.1016/S0020-0190(02)00346-0 (see page 37).

[KS16] Timo Kötzing and Martin Schirneck. **Towards an Atlas of Computational Learning Theory**. In: *Proceedings of the 33rd Symposium on Theoretical Aspects of Computer Science (STACS)*. 2016, 47:1–47:13. DOI: 10.4230/LIPIcs.STACS.2016.47 (see page x).

[KS21] Timo Kötzing and Karen Seidel. **Learning Languages in the Limit from Positive Information with Finitely Many Memory Changes**. In: *Proceedings of the 17th Conference on Computability in Europe (CiE)*. 2021, 318–329. DOI: 10.1007/978-3-030-80049-9_29 (see page x).

[KSS17] Timo Kötzing, Martin Schirneck, and Karen Seidel. **Normal Forms in Semantic Language Identification**. In: *Proceedings of the 28th International Conference on Algorithmic Learning Theory (ALT)*. 2017, 493–516. URL: http://proceedings.mlr.press/v76/k%C3%B6tzing17a (see page x).

[KV94] Michael J. Kearns and Umesh V. Vazirani. **An Introduction to Computational Learning Theory**. Cambridge, MA, USA: MIT Press, 1994 (see page 92).

[KW18] Daniel M. Kane and R. Ryan Williams. **The Orthogonal Vectors Conjecture for Branching Programs and Formulas**. In: *Proceedings of the 10th Innovations in Theoretical Computer Science Conference (ITCS)*. 2018, 48:1–48:15. DOI: 10.4230/LIPIcs.ITCS.2019.48 (see page 111).

[Law72] Eugene L. Lawler. **A Procedure for Computing the *K* Best Solutions to Discrete Optimization Problems and Its Application to the Shortest Path Problem**. *Management Science* 18 (1972), 401–405. URL: https://www.jstor.org/stable/2629357 (see page 60).

[Lit69] John E. Littlewood. **On the Probability in the Tail of a Binomial Distribution**. *Advances in Applied Probability* 1 (1969), 43–72. DOI: 10.2307/1426408 (see page 93).

[Liv+20] Ester Livshits, Alireza Heidari, Ihab F. Ilyas, and Benny Kimelfeld. **Approximate Denial Constraints**. *Proceedings of the VLDB Endowment* 13 (2020), 1682–1695. DOI: 10.14778/3401960.3401966 (see pages 161, 164).

[LY99] Elliott H. Lieb and Jakob Yngvason. **The Physics and Mathematics of the Second Law of Thermodynamics**. *Physics Reports* 310 (1999), 1–96. ISSN: 0370-1573. DOI: 10.1016/S0370-1573(98)00082-9 (see page 109).

[Mai83] David Maier. **The Theory of Relational Databases**. Rockville, MD, USA: Computer Science Press, 1983 (see page 28).

[Man16] Christian Mancas. **Algorithms for Database Keys Discovery Assistance**. In: *Proceedings of the 15th International Conference on Perspectives in Business Informatics Research (BIR)*. 2016, 322–338 (see page 134).

[Mar13] Arnaud Mary. **Énumération des Dominants Minimaux d'un Graphe**. Université Blaise Pascal, PhD thesis. (Enumeration of Minimal Dominating Sets in Graphs.) In French. 2013 (see page 64).

[Mck89] Brendan D. Mckay. **On Littlewood's Estimate for the Binomial Distribution**. *Advances in Applied Probability* 21 (1989), 475–478. DOI: 10.2307/1427172 (see page 93).

[Mei20] Arne Meier. **Parametrised Enumeration**. Leibniz University Hannover, Habilitationsschrift. 2020. DOI: 10.15488/9427 (see page 4).

[MR87] Heikki Mannila and Kari-Jouko Räihä. **Dependency Inference**. In: *Proceedings of the 13th International Conference on Very Large Data Bases (VLDB)*. 1987, 155–158. URL: https://www.vldb.org/conf/1987/P155.PDF (see pages 2, 13, 26, 132, 135).

[MR94] Heikki Mannila and Kari-Jouko Räihä. **Algorithms for Inferring Functional Dependencies from Relations**. *Data & Knowledge Engineering* 12 (1994), 83–99 (see page 134).

[MR95] Rajeev Motwani and Prabhakar Raghavan. **Randomized Algorithms**. Cambridge, UK: Cambridge University Press, 1995. DOI: 10.1017/CBO9780511814075 (see pages 9, 22, 92).

[MS19] Arnaud Mary and Yann Strozecki. **Efficient Enumeration of Solutions Produced by Closure Operations**. *Discrete Mathematics & Theoretical Computer Science* 21 (2019). DOI: 10.23638/DMTCS-21-3-22 (see page 60).

[MU14] Keisuke Murakami and Takeaki Uno. **Efficient Algorithms for Dualizing Large-Scale Hypergraphs**. *Discrete Applied Mathematics* 170 (2014), 83–94. DOI: 10.1016/j.dam.2014.01.012 (see pages 3, 11, 38, 132, 135, 136, 139, 142, 151, 153).

[MU17] Michael Mitzenmacher and Eli Upfal. **Probability and Computing**. 2nd Edition. New York City, NY, USA: Cambridge University Press, 2017. DOI: 10.1017/CBO9780511813603 (see pages 9, 92).

[Muj+20] Felix Mujkanovic, Vanja Doskoč, Martin Schirneck, Patrick Schäfer, and Tobias Friedrich. **timeXplain - A Framework for Explaining the Predictions of Time Series Classifiers**. *CoRR* abs/2007.07606 (2020). ArXiv preprint. arXiv: 2007.07606 [cs.LG]. URL: https://arxiv.org/abs/2007.07606 (see page x).

[NC10] Michael A. Nielsen and Isaac L. Chuang. **Quantum Computation and Quantum Information: 10th Anniversary Edition**. Cambridge, UK: Cambridge University Press, 2010. DOI: 10.1017/CBO9780511976667 (see page 110).

[New01] Mark E. J. Newman. **Scientific Collaboration Networks. I. Network Construction and Fundamental Results**. *Physical Review E* 64 (2001). Article no. 016131. DOI: 10.1103/PhysRevE.64.016131 (see page 110).

[New10] Mark E. J. Newman. **Networks: An Introduction**. New York City, NY, USA: Oxford University Press, 2010. DOI: 10.1093/acprof:oso/9780199206650.001.0001 (see page 110).

[Nie06] Rolf Niedermeier. **Invitation to Fixed-Parameter Algorithms**. Oxford Lecture Series in Mathematics and Its Applications. Oxford, UK: Oxford University Press, 2006. DOI: 10.1093/acprof:oso/9780198566076.001.0001 (see pages 16, 24).

[ODo14] Ryan O'Donnell. **Analysis of Boolean Functions**. Cambridge, UK: Cambridge University Press, 2014. DOI: 10.1017/CBO9781139814782 (see page 111).

[OW15] Pietro S. Oliveto and Carsten Witt. **Improved Time Complexity Analysis of the Simple Genetic Algorithm**. *Theoretical Computer Science* 605 (2015), 21–41. DOI: 10.1016/j.tcs.2015.01.002 (see pages 103, 104).

[Pap+15] Thorsten Papenbrock, Jens Ehrlich, Jannik Marten, Tommy Neubert, Jan-Peer Rudolph, Martin Schönberg, Jakob Zwiener, and Felix Naumann. **Functional Dependency Discovery: An Experimental Evaluation of Seven Algorithms**. *Proceedings of the VLDB Endowment* 8 (2015), 1082–1093. DOI: 10.14778/2794367.2794377 (see pages 7, 55, 82).

[Paw91] Zdzisław I. Pawlak. **Rough Sets: Theoretical Aspects of Reasoning About Data**. Vol. 9. Theory and Decision Library - Series D: System Theory, Knowledge Engineering and Problem Solving. Dordrecht, Netherlands: Kluwer Academic Publishers, 1991. DOI: 10.1007/978-94-011-3534-4 (see page 30).

[PN04] Juyong Park and Mark E. J. Newman. **The Statistical Mechanics of Networks**. *Physical Review E* 70 (2004). Article no. 066117. DOI: 10.1103/PhysRevE.70.066117 (see page 110).

[PN16] Thorsten Papenbrock and Felix Naumann. **A Hybrid Approach to Functional Dependency Discovery**. In: *Proceedings of the 2016 International Conference on Management of Data (SIGMOD)*. 2016, 821–833. DOI: 10.1145/2882903.2915203 (see pages 7, 26, 135, 140, 141, 143, 161).

[PN17] Thorsten Papenbrock and Felix Naumann. **A Hybrid Approach for Efficient Unique Column Combination Discovery**. In: *Proceedings of the 17th Fachtagung Datenbanksysteme in Business, Technologie und Web Technik (BTW)*. 2017, 195–204. URL: https://dl.gi.de/20.500.12116/628 (see pages 6, 7, 13, 90, 131, 134, 138, 140, 150, 153, 164).

[Pro53] Yuri V. Prokhorov. **Asimptoticheskoe povedenie binomial'nogo raspredelenija**. *Uspekhi Matematicheskikh Nauk* 8 (1953). (Asymptotic Behavior of the Binomial Distribution.) In Russian, 135–142. URL: http://mi.mathnet.ru/eng/umn8214 (see page 93).

[PS94] Uri N. Peled and Bruno Simeone. **A $O(nm)$-Time Algorithm for Computing the Dual of a Regular Boolean Function**. *Discrete Applied Mathematics* 49 (1994), 309–323. DOI: 10.1016/0166-218X(94)90215-1 (see page 37).

[Pus+20] Taneli Pusa, Mariana Galvao Ferrarini, Ricardo Andrade, Arnaud Mary, Alberto Marchetti-Spaccamela, Leen Stougie, and Marie-France Sagot. **MOOMIN - Mathematical explOration of 'Omics data on a MetabolIc Network**. *Bioinformatics* 36 (2020), 514–523. DOI: 10.1093/bioinformatics/btz584 (see page 26).

[Rei87] Raymond Reiter. **A Theory of Diagnosis from First Principles**. *Artificial Intelligence* 32 (1987), 57–95. DOI: 10.1016/0004-3702(87)90062-2 (see pages 2, 26, 135).

[RT75] Ronald C. Read and Robert E. Tarjan. **Bounds on Backtrack Algorithms for Listing Cycles, Paths, and Spanning Trees**. *Networks* 5 (1975), 237–252. DOI: 10.1002/net.1975.5.3.237 (see page 60).

[Sar+15] Fabio Saracco, Riccardo Di Clemente, Andrea Gabrielli, and Tiziano Squartini. **Randomizing Bipartite Networks: The Case of the World Trade Web**. *Scientific Reports* 5 (2015), Art. no. 10595. DOI: 10.1038/srep10595 (see page 110).

[SGI19] Hemant Saxena, Lukasz Golab, and Ihab F. Ilyas. **Distributed Implementations of Dependency Discovery Algorithms**. *Proceedings of the VLDB Endowment* 12 (2019), 1624–1636. DOI: 10.14778/3342263.3342638 (see page 135).

[Sha48] Claude E. Shannon. **A Mathematical Theory of Communication**. *The Bell System Technical Journal* 27 (1948), 379–423. DOI: 10.1002/j.1538-7305.1948.tb01338.x (see page 109).

[Shi+17] Feng Shi, Martin Schirneck, Tobias Friedrich, Timo Kötzing, and Frank Neumann. **Reoptimization Times of Evolutionary Algorithms on Linear Functions Under Dynamic Uniform Constraints**. In: *Proceedings of the 2017 Genetic and Evolutionary Computation Conference (GECCO)*. 2017, 1407–1414. DOI: 10.1145/3071178.3071270 (see page x).

[Shi+19] Feng Shi, Martin Schirneck, Tobias Friedrich, Timo Kötzing, and Frank Neumann. **Reoptimization Time Analysis of Evolutionary Algorithms on Linear Functions Under Dynamic Uniform Constraints**. *Algorithmica* 81 (2019), 828–857. DOI: 10.1007/s00453-018-0451-4 (see page x).

[Shi+20] Feng Shi, Martin Schirneck, Tobias Friedrich, Timo Kötzing, and Frank Neumann. **Correction to: Reoptimization Time Analysis of Evolutionary Algorithms on Linear Functions Under Dynamic Uniform Constraints**. *Algorithmica* 82 (2020), 3117–3123. DOI: 10.1007/s00453-020-00739-x (see page x).

[Sis+06] Yannis Sismanis, Paul Brown, Peter J. Haas, and Berthold Reinwald. **GORDIAN: Efficient and Scalable Discovery of Composite Keys**. In: *Proceedings of the 32nd International Conference on Very Large Data Bases (VLDB)*. 2006, 691–702. URL: https://www.vldb.org/conf/2006/p691-sismanis.pdf (see pages 13, 131, 134).

[Slu77] Eric V. Slud. **Distribution Inequalities for the Binomial Law**. *Annals of Probability* 5 (1977), 404–412. DOI: 10.1214/aop/1176995801 (see page 93).

[SR92] Andrzej Skowron and Cecylia Rauszer. **The Discernibility Matrices and Functions in Information Systems**, 331–362. In: *Intelligent Decision Support: Handbook of Applications and Advances of the Rough Sets Theory*. Ed. by Roman Słowiński. Vol. 11. Theory and Decision Library - Series D: System Theory, Knowledge Engineering and Problem Solving. Dordrecht, Netherlands: Kluwer Academic Publishers, 1992. DOI: 10.1007/978-94-015-7975-9_21 (see page 30).

[SS85] Jeanette Schmidt-Pruzan and Eli Shamir. **Component Structure in the Evolution of Random Hypergraphs**. *Combinatorica* 5 (1985), 81–94. DOI: 10.1007/BF02579445 (see page 110).

[Str13] Yann Strozecki. **On Enumerating Monomials and Other Combinatorial Structures by Polynomial Interpolation**. *Theory of Computing Systems* 53 (2013), 532–568. DOI: 10.1007/s00224-012-9442-z (see page 15).

[Swa02] Aaron Swartz. **MusicBrainz: A Semantic Web Service**. *IEEE Intelligent Systems* 17 (2002). See `www.musicbrainz.org`, 76–77. DOI: 10.1109/5254.988466 (see page 25).

[Vas19] Virginia Vassilevska Williams. **On Some Fine-Grained Questions in Algorithms and Complexity**. In: *Proceedings of the 2018 International Congress of Mathematicians (ICM)*. 2019, 3447–3487. DOI: 10.1142/9789813272880_0188 (see pages 5, 71).

[VW18] Virginia Vassilevska Williams and R. Ryan Williams. **Subcubic Equivalences Between Path, Matrix, and Triangle Problems**. *Journal of the ACM* 65 (2018), Art. no. 27, 1–38. DOI: 10.1145/3186893 (see page 19).

[Weg91] Ingo Wegener. **The Complexity of Boolean Functions**. Vol. 3. Applicable Theory in Computer Science. Hoboken, NJ, USA: Wiley-Teubner, 1991. URL: http://ls2-www.cs.uni-dortmund.de/monographs/bluebook/ (see page 47).

[WGR01] Catharine M. Wyss, Chris Giannella, and Edward L. Robertson. **FastFDs: A Heuristic-Driven, Depth-First Algorithm for Mining Functional Dependencies from Relation Instances**. In: *Proceedings of the 13th International Conference of Data Warehousing and Knowledge Discovery (DaWaK)*. 2001, 101–110. DOI: 10.1007/3-540-44801-2_11 (see page 135).

[Wil05] R. Ryan Williams. **A New Algorithm for Optimal 2-Constraint Satisfaction and Its Implications**. *Theoretical Computer Science* 348 (2005), 357–365. DOI: 10.1016/j.tcs.2005.09.023 (see pages 19, 72, 73, 111).

[Wil16] R. Ryan Williams. **Strong ETH Breaks With Merlin and Arthur: Short Non-Interactive Proofs of Batch Evaluation**. In: *Proceedings of the 31st Conference on Computational Complexity (CCC)*. 2016, 2:1–2:17. DOI: 10.4230/LIPIcs.CCC.2016.2 (see page 75).

[WLL19] Ziheng Wei, Uwe Leck, and Sebastian Link. **Discovery and Ranking of Embedded Uniqueness Constraints**. *Proceedings of the VLDB Endowment* 12 (2019), 2339–2352. DOI: 10.14778/3358701.3358703 (see pages 13, 26, 134, 161).

[WY13] Oren Weimann and Raphael Yuster. **Replacement Paths and Distance Sensitivity Oracles via Fast Matrix Multiplication**. *ACM Transactions on Algorithms* 9 (2013), 14:1–14:13. DOI: 10.1145/2438645.2438646 (see page 165).

[Zhu+19] Guanghui Zhu, Qian Wang, Qiwei Tang, Rong Gu, Chunfeng Yuan, and Yihua Huang. **Efficient and Scalable Functional Dependency Discovery on Distributed Data-Parallel Platforms**. *IEEE Transactions on Parallel and Distributed Systems* 30 (2019), 2663–2676. DOI: 10.1109/TPDS.2019.2925014 (see page 135).

[Zwe14] Katharina A. Zweig. **Network Analysis Literacy: A Practical Approach to the Analysis of Networks**. Lecture Notes in Social Networks. Vienna, Austria: Springer, 2014. DOI: 10.1007/978-3-7091-0741-6 (see page 110).

List of Publications

Articles in Refereed Journals

[1] **Efficiently Enumerating Hitting Sets of Hypergraphs Arising in Data Profiling**. *Journal of Computer and System Sciences* 124 (2022), 192–213. DOI: 10.1016/j.jcss.2021.10.002. Joint work with Thomas Bläsius, Tobias Friedrich, Julius Lischeid, and Kitty Meeks.

[2] **The Complexity of Dependency Detection and Discovery in Relational Databases**. *Theoretical Computer Science* 900 (2022), 79–96. Joint work with Thomas Bläsius and Tobias Friedrich.

[3] **Hitting Set Enumeration with Partial Information for Unique Column Combination Discovery**. *Proceedings of the VLDB Endowment* 13 (2020), 2270–2283. DOI: 10.14778/3407790.3407824. Joint work with Johann Birnick, Thomas Bläsius, Tobias Friedrich, Felix Naumann, and Thorsten Papenbrock.

[4] **Analysis of the (1+1) EA on Subclasses of Linear Functions under Uniform and Linear Constraints**. *Theoretical Computer Science* 832 (2020), 3–19. DOI: 10.1016/j.tcs.2018.04.051. Joint work with Tobias Friedrich, Timo Kötzing, J. A. Gregor Lagodzinski, and Frank Neumann.

[5] **Correction to: Reoptimization Time Analysis of Evolutionary Algorithms on Linear Functions Under Dynamic Uniform Constraints**. *Algorithmica* 82 (2020), 3117–3123. DOI: 10.1007/s00453-020-00739-x. Joint work with Feng Shi, Tobias Friedrich, Timo Kötzing, and Frank Neumann.

[6] **Reoptimization Time Analysis of Evolutionary Algorithms on Linear Functions Under Dynamic Uniform Constraints**. *Algorithmica* 81 (2019), 828–857. DOI: 10.1007/s00453-018-0451-4. Joint work with Feng Shi, Tobias Friedrich, Timo Kötzing, and Frank Neumann.

[7] **Island Models Meet Rumor Spreading**. *Algorithmica* 81 (2019), 886–915. DOI: 10.1007/s00453-018-0445-2. Joint work with Benjamin Doerr, Philipp Fischbeck, Clemens Frahnow, Tobias Friedrich, and Timo Kötzing.

Articles in Refereed Conference Proceedings

[8] **Deterministic Sensitivity Oracles for Diameter, Eccentricities and All Pairs Distances**. In: *Proceedings of the 49th International Colloquium on Automata, Languages, and Programming (ICALP)*. 2022, 22:1–22:19. DOI: 10.4230/LIPIcs.ICALP.2022.22. Joint work with Davide Bilò, Keerti Choudhary, Sarel Cohen, and Tobias Friedrich.

[9] **Crossover for Cardinality Constrained Optimization**. In: *Proceedings of the 2022 Genetic and Evolutionary Computation Conference (GECCO)*. 2022, 1399–1407. DOI: 10.1145/3512290.3528713. Joint work with Tobias Friedrich, Timo Kötzing, Aishwarya Radhakrishnan, Leon Schiller, Georg Tennigkeit, and Simon Wietheger.

[10] **Fixed-Parameter Sensitivity Oracles**. In: *Proceedings of the 13th Innovations in Theoretical Computer Science Conference (ITCS)*. 2022, 23:1–23:18. DOI: 10.4230/LIPIcs.ITCS.2022.23. Joint work with Davide Bilò, Katrin Casel, Keerti Choudhary, Sarel Cohen, Tobias Friedrich, J.A. Gregor Lagodzinski, and Simon Wietheger.

[11] **Near-Optimal Deterministic Single-Source Distance Sensitivity Oracles**. In: *Proceedings of the 29th European Symposium on Algorithms (ESA)*. 2021, 18:1–18:17. DOI: 10.4230/LIPIcs.ESA.2021.18. Joint work with Davide Bilò, Sarel Cohen, and Tobias Friedrich.

[12] **Space-Efficient Fault-Tolerant Diameter Oracles**. In: *Proceedings of the 46th International Symposium on Mathematical Foundations of Computer Science (MFCS)*. 2021, 18:1–18:16. DOI: 10.4230/LIPIcs.MFCS.2021.18. Joint work with Davide Bilò, Sarel Cohen, and Tobias Friedrich.

[13] **The Minimization of Random Hypergraphs**. In: *Proceedings of the 28th European Symposium on Algorithms (ESA)*. 2020, 21:1–21:15. DOI: 10.4230/LIPIcs.ESA.2020.21. Joint work with Thomas Bläsius and Tobias Friedrich.

[14] **Understanding the Effectiveness of Data Reduction in Public Transportation Networks**. In: *Proceedings of the 16th Workshop on Algorithms and Models for the Web Graph (WAW)*. 2019, 87–101. DOI: 10.1007/978-3-030-25070-6_7. Joint work with Thomas Bläsius, Philipp Fischbeck, and Tobias Friedrich.

[15] **Efficiently Enumerating Hitting Sets of Hypergraphs Arising in Data Profiling**. In: *Proceedings of the 21st Meeting on Algorithm Engineering and Experiments (ALENEX)*. 2019, 130–143. DOI: 10.1137/1.9781611975499.11. Joint work with Thomas Bläsius, Tobias Friedrich, Julius Lischeid, and Kitty Meeks.

[16] **Normal Forms in Semantic Language Identification**. In: *Proceedings of the 28th International Conference on Algorithmic Learning Theory (ALT)*. 2017, 493–516. URL: http://proceedings.mlr.press/v76/k%C3%B6tzing17a.html. Joint work with Timo Kötzing and Karen Seidel.

[17] **Reoptimization Times of Evolutionary Algorithms on Linear Functions Under Dynamic Uniform Constraints**. In: *Proceedings of the 2017 Genetic and Evolutionary Computation Conference (GECCO)*. 2017, 1407–1414. DOI: 10.1145/3071178.3071270. Joint work with Feng Shi, Tobias Friedrich, Timo Kötzing, and Frank Neumann.

[18] **Island Models Meet Rumor Spreading**. In: *Proceedings of the 2017 Genetic and Evolutionary Computation Conference (GECCO)*. 2017, 1359–1366. DOI: 10.1145/3071178.3071206. Joint work with Benjamin Doerr, Philipp Fischbeck, Clemens Frahnow, Tobias Friedrich, and Timo Kötzing.

[19] **Analysis of the (1+1) EA on Subclasses of Linear Functions under Uniform and Linear Constraints**. In: *Proceedings of the 14th Conference on Foundations of Genetic Algorithms (FOGA)*. 2017, 45–54. DOI: 10.1145/3040718.3040728. Joint work with Tobias Friedrich, Timo Kötzing, J. A. Gregor Lagodzinski, and Frank Neumann.

[20] **The Parameterized Complexity of Dependency Detection in Relational Databases** . In: *Proceedings of the 11th International Symposium on Parameterized and Exact Computation (IPEC)*. 2016, 6:1–6:13. DOI: 10.4230/LIPIcs.IPEC.2016.6. Joint work with Thomas Bläsius and Tobias Friedrich.

[21] **Fast Building Block Assembly by Majority Vote Crossover**. In: *Proceedings of the 2016 Genetic and Evolutionary Computation Conference (GECCO)*. 2016, 661–668. DOI: 10.1145/2908812.2908884. Joint work with Tobias Friedrich, Timo Kötzing, Martin S. Krejca, Samadhi Nallaperuma, and Frank Neumann.

[22] **Towards an Atlas of Computational Learning Theory**. In: *Proceedings of the 33rd Symposium on Theoretical Aspects of Computer Science (STACS)*. 2016, 47:1–47:13. DOI: 10.4230/LIPIcs.STACS.2016.47. Joint work with Timo Kötzing.