Polynomially Solvable Cases of Hypergraph Transversal and Related Problems

Imran Rauf

Thesis for obtaining the title of Doctor of Engineering

of the Faculties of Natural Sciences and Technology

of Saarland University

Saarbrücken, Germany, Oct 2011

Abstract

This thesis is mainly concerned with the *hypergraph transversal problem*, which asks to generate all minimal transversals of a given hypergraph. While the current best upper bound on the complexity of the problem is quasi-polynomial in the combined input and output sizes, it is shown to be solvable in output polynomial time for a number of hypergraph classes. We extend this polynomial frontier to the hypergraphs induced by hyperplanes and constant-sided polytopes in fixed dimension $\mathbb{R}^d$ and hypergraphs for which every minimal transversal and hyperedge intersection is bounded. We also show the problem to be fixed parameter tractable with respect to the minimum integer k such that the input hypergraph is k-degenerate, and also with respect to its maximum complementary degree. Whereas we improve the known bounds when the parameter is the maximum degree of a hypergraph.

We also study the *readability* of a monotone Boolean function which is defined as the minimum integer r such that it can be represented by a $\vee$-$\wedge$-formula with every variable occurrence is bounded by r. We prove that it is NP-hard to approximate the readability of even a depth three Boolean formula. We also give tight sublinear upper bounds on the readability of a monotone Boolean function given in CNF (or DNF) form, parameterized by the number of terms in the CNF and the maximum number of variables in the intersection of any constant number of terms. For interval DNF's we give much tighter logarithmic bounds on the readability.

Finally, we discuss an implementation of a quasi-polynomial algorithm for the hypergraph transversal problem that runs in polynomial space. We found our implementation to be competitive with all but one previous implementation on various datasets.

Kurzfassung

Diese Dissertation behandelt hauptsächlich das Transversalhypergraphen-Problem: gegeben ein Hypergraph, generiere alle seine minimalen Transversalen. Obwohl die bisher beste obere Schranke für die Komplexität dieses Problems quasi-polynomiell in der Größe der Ein- und Ausgabe ist, sind für viele Klassen von Hypergraphen output-polynomielle Lösungen bekannt. Wir zeigen fr zwei weitere Klassen von Hypergraphen, dass sie output-polynomielle Lösungen besitzen. Zum einen sind dies Hypergraphen, die durch Hyperebenen und Polytope mit konstanter Seitenzahl im $\mathbb{R}^d$ für eine feste Dimension d induziert werden; zum anderen sind dies Hypergraphen, für die die Größe der Schnittmenge jedes Transversalen und jeder Hyperkante beschränkt ist. Desweiteren zeigen wir, dass das Problem fixed-parameter tractable ist, wenn als Parameter der maximale inverse Knotengrad des eingegebenen Hypergraphen gewählt wird oder der Parameter k die kleinste natürliche Zahl ist, für die der eingegebene Hypergraph k-degeneriert ist. Wir verbessern die bekannten fixed-parameter Ergebnisse für den Parameter des maximalen Knotengrads des eingegebenen Hypergraphen.

Außerdem untersuchen wir die *readability* von monotonen Booleschen Funktionen. Diese ist als kleinste natürliche Zahl r definiert, so dass die Funktion als $\vee$-$\wedge$-Formel repräsentiert werden kann, in der jede Variable höchstens r-mal vorkommt. Wir beweisen, dass die Approximation der readability schon für Boolesche Formeln der Tiefe 3 NP-hart ist. Darüberhinaus geben wir scharfe sublineare untere Schranken für die readability monotoner Boolescher Funktionen an, die in KNF-Form (oder DNF-Form) gegeben sind, wenn über die Anzahl der Terme der KNF und die maximale Anzahl an Variablen im Durchschnitt einer konstanten Anzahl von Termen parametrisiert wird. Für intervall-DNF geben wir noch schärfere logarithmische Schranken für die readability an.

Schließlich behandeln wir für das Transversalhypergraphen-Problem noch die Implementation eines quasi-polynomiellen Algorithmus, der mit polynomiellen Platzbedarf auskommt. Auf verschiedenen Datensätzen ist unsere Implementation zu allen vorherigen Implementationen (außer einer) konkurrenzfähig.

Acknowledgements

This work would have been impossible without the much needed encouragement and energetic input from my advisor Khaled Elbassioni throughout its development and completion, for which I am really grateful to him. I am indebted to Kurt Mehlhorn, the director of Algorithm and Complexity group at the Max Planck Institute of Computer Science (MPII), for giving me chance to pursue PhD in his excellent group, where he is instrumental in maintaining a great atmosphere for research and also provides a role model for younger members of the group. I am also grateful to Ulrich Meyer and Stefan Funke for guiding me in the earlier phase of my PhD and Prof. Endre Boros for agreeing to examine my thesis. I was fortunate to receive the generous stipend awarded by International Max Planck Research School for Computer Science which supported me throughout my time at MPII. I would like to thank my coauthors Khaled Elbassioni, Stefan Funke, Kaz Makino, Matthias Hagen and Saurabh Ray. Matthias also did a great favor by translating the abstract into German.

My colleagues and friends helped me immensely in some way or another while I was working on this thesis. Here I would like to acknowledge some names that come to mind now: Tobias and Evangelia for being great officemates; Deepak, Hans, Saurabh and Vikram for being good friends and inspiring colleagues; Amir, Anuj, Kara, Awais, Habeeb and Sikander for many fun activities we did together; and Safdar, Sohail, Atif, Mujtaba, Usman, Yasir, Jamal, Naveed, Shahid, Khurram, Hameed and Fawad for countless political/religious discussions while enjoying delicious Pakistani food together.

Finally, I cannot express my gratitude enough for my parents Abdul Rauf and Mussarat-un-Nissa, and my siblings Adnan, Saba, Salman and Masoom. It was my elder brother Adnan who introduced me to Computer Science in the first place. This thesis is dedicated to my wife Saeeda and our sweet daughter Manal.

Contents

Chapter 1

Introduction

Monotone enumeration problems, which call for finding all objects or configurations satisfying a certain *monotone property*, are often captured by the *hypergraph transversal problem*. Although transversal hypergraphs have been studied before by mathematicians, Johnson, Papadimitriou and Yannakakis [JPY88] first proposed it as a computation problem. Currently, the fastest known algorithm [FK96] for solving the hypergraph transversal problem runs in time quasi-polynomial in the combined input and output size. In this chapter we give a brief overview of the problem. We then give a summary of the work presented in the thesis.

1.1 Previous Work

The *hypergraph transversal problem* asks to generate all minimal *transversals* (hitting-sets) of a given hypergraph. The family of all minimal transversals, which itself is a hypergraph, is called *transversal hypergraph*. This problem has received considerable attention in the literature (see, e. g., [BI95, EG95, EGM03, Got04, Lov92, Pap97]), since it is known to be polynomially or quasi-polynomially equivalent with many problems in various areas, such as artificial intelligence (e. g., [EG95, KPS93]), database theory (e. g., [MR86]), distributed systems (e. g., [GMB85, IK93]), machine learning and data mining (e. g., [AB92, BGKM02, GMKT97]), mathematical programming (e. g., [BEG^{+}02, Kha00]), matroid theory (e. g., [KBE+05]), and reliability theory (e. g., [Col87, Ram90]) to name a few.

As the number of minimal transversals of a hypergraph can be exponential in the size of the input hypergraph, one can only hope for an algorithm whose efficiency is measured in terms of combined input and output sizes. The currently fastest known algorithm [FK96] for solving the hypergraph transversal problem runs in quasi-polynomial time $N^{o(\log N)}$, where N is the combined input and output size. Several quasi-polynomial time algorithms with some other desirable properties also exist. A parallel algorithm that runs in polylogarithmic time using quasi-polynomial number of processors is recently given in [Elb08, BM09]. Other algorithms that match the current best quasi-polynomial bound and run in polynomial space are presented in [Tam00, Elb08]. An algorithm that use only multiplication is shown to achieve quasi-polynomial time in [Elb06]. Yet another quasi-polynomial time algorithm for a variant of the hypergraph transversal problem is analyzed with respect to its average case complexity in [GK04]. The decision version of the problem is known to be solvable with limited nondeterminism [KS03], i. e., by nondeterministic polynomial time algorithm that makes only polylogarithmic number of guesses.

While it is still open whether the problem can be solved in output polynomial time for arbitrary hypergraphs, output polynomial time algorithms exist for several classes of hypergraphs, e. g., hypergraphs of bounded edge-size [BEGK04, EG95], of bounded degree [DMP99, EGM03], of bounded-edge intersections [BEGK04], of bounded conformality [BEGK04], of bounded treewidth [EGM03], of bounded latency [MI97], and read-once (exact) hypergraphs [Eit94].

1.2 Summary and Outline of the Thesis

The basic definitions and notations are introduced in Chapter 2, which also briefly presents the results about the hypergraph transversal problem relevant to this thesis.

We then proceed in the following three chapters with showing the output polynomial time solvability of the hypergraph transversal problem for the following classes of hypergraphs.

Fixed Parameter Tractability (Chapter 3): We present FPT algorithms for the hypergraph transversal problem with the number of edges of the hypergraphs, the minimum integer k such that the input hypergraph is k-degenerate, and the vertex complementary degree as our parameters. Moreover, we also get FPT results for the hypergraph transversal problem as well as the related problems of generating all maximal independent sets of a hypergraph and all maximal frequent sets where parameters bound the intersections or unions of edges.

r-Exact Hypergraphs (Chapter 4): We call a hypergraph $\mathcal{H}$ is r-exact for a positive integer r, if any minimal transversal of $\mathcal{H}$ intersects any hyperedge of $\mathcal{H}$ in at most r vertices. This class includes several interesting examples from geometry, e.g., circular-arc hypergraphs ($r = 2$), hypergraphs defined by sets of axis-parallel lines stabbing a given set of α-fat objects ($r = 4\alpha$), and hypergraphs defined by sets of points contained in translates of a given cone in the plane ($r = 2$). For constant r, we give a polynomial-time algorithm to decide for a pair of r-exact hypergraphs, if one is the transversal hypergraph of the other. This result implies that minimal hitting sets for the above geometric hypergraphs can be generated in output polynomial time.

Geometric Hypergraphs (Chapter 5): For hypergraphs induced by intersections of a family of geometrical objects by another, we introduce two general frameworks to generate the transversal hypergraph. The first approach use only elementary techniques, and gives a polynomial-time algorithm for the problem of hitting hyper-rectangles by points, and stabbing connected objects by axis-parallel hyperplanes, both in $\mathbb{R}^d$ for a fixed d. Overcoming the limitations of the first approach, we give another technique that use simplicial partitions and cuttings to efficiently enumerate all minimal hitting-sets as well as minimal covers of hypergraphs defined by intersections of sets of points with hyperplanes (and hence balls) or more generally, constant-sided polytopes in fixed dimension $\mathbb{R}^d$. Finally, we give a polynomial delay algorithm for the special case of hypergraphs induced by half-planes and points in $\mathbb{R}^2$.

In the second half of the thesis, we begin with a study of a complexity measure for monotone Boolean functions called *readability* in Chapter 6. The readability of a monotone Boolean function $f : \{0, 1\}^n \to \{0, 1\}$ is defined to be the minimum integer k such that there exists an $\wedge$-$\vee$-formula equivalent to f in which each variable appears at most k times. An important open problem in this area is whether there exists a polynomial-time algorithm, which given a monotone Boolean function f, in CNF or DNF form, checks whether f is a read-k function, for a fixed k. We answer a related question already for $k = 2$ by showing that it is NP-hard to decide if a given monotone formula represents a read-twice function. It follows also from our reduction that it is NP-hard to approximate the readability of a given monotone Boolean function f within a factor of $\mathcal{O}(n)$. We also give tight sublinear upper bounds on the readability of a monotone Boolean function given in CNF (or DNF) form, parameterized by the number of terms in the CNF and the maximum size in each term, or more generally by the maximum number of variables in the intersection of any constant number of terms. When the variables of the DNF can be ordered so that each term consists of a set of consecutive variables, we give much tighter logarithmic bounds on readability.

Finally, we discuss an implementation of a polynomial space algorithm of Elbassioni [Elb08] for the hypergraph transversal problem in Chapter 7. The distinguishing feature of our implementation is that it requires polynomial space with the same bound on the running time as the current best. In contrast, all of the previous implementations either have exponential worst-case running time or need super-polynomial space. We found our implementation to be competitive with all but one previous implementation on various datasets.

Chapter 2

Preliminaries

2.1 Background and Notation

A hypergraph is a pair $(V, \mathcal{H})$ where $V = [n] \overset{\text{def}}{=} \{1, 2, \ldots, n\}$ is a ground set and members of $\mathcal{H}$ are subsets of V, i. e., $\mathcal{H} \subseteq 2^V$. Hypergraphs can be viewed as generalizations of graphs and in that respect, elements of V and $\mathcal{H}$ are called *vertices* and *hyperedges*, respectively. When the set of vertices V is clear from the context, we only refer to $\mathcal{H}$ as our hypergraph for notational convenience and denote by $V(\mathcal{H})$ the vertex set of $\mathcal{H}$. We also sometimes omit the prefix *hyper* and refer to the the elements of $\mathcal{H}$ as simply *edges*. A hypergraph $\mathcal{H}$ is called *trivial* when $\mathcal{H}$ is empty.

A hypergraph $\mathcal{H} = \{H_1, \ldots, H_m\}$ is *Sperner* when for any two hyperedges H_i, H_j of $\mathcal{H}$, $H_i \subseteq H_j$ implies $i = j$, i. e., when no hyperedge contains another. For any hypergraph $\mathcal{H} \subseteq 2^V$, let us denote by $\min(\mathcal{H})$ the Sperner hypergraph we get after deleting the inclusion-wise non-minimal hyperedges in $\mathcal{H}$.

A vertex $u \in V$ *hits* a hyperedge $H \in \mathcal{H}$ when $u \in H$. A subset of vertices $X \in V$ is called a *transversal* (or a *hitting-set*) of the hypergraph $\mathcal{H} \subseteq 2^V$ when every hyperedge of $\mathcal{H}$ is hit by some element in X, i. e., $X \cap H \neq \emptyset$, $\forall H \in \mathcal{H}$. A transversal is *minimal* when any of its proper subsets is not a transversal. The hypergraph consisting of all minimal transversals of $\mathcal{H}$ is called the *transversal hypergraph* of $\mathcal{H}$ and is denoted by

$\mathcal{H}^d$. Note that $\mathcal{H}^d$ is Sperner by definition. Also, it is well known that $(\mathcal{H}^d)^d = \min(\mathcal{H})$ (see Section 2.2 for a proof) and so $\mathcal{H}^d$ is also called the *dual* hypergraph of $\mathcal{H}$.

Example 2.1 (Matching). *Let our hypergraph be $\mathcal{H} = \{\{i, n+i\} \mid i = 1, \ldots, n\}$, where n is a positive integer. Then its dual hypergraph is $\mathcal{H}^d = \{\{j_1, j_2, \ldots, j_n\} \mid j_i \in \{i, n+i\}\}$.*

Example 2.2 (Bipartite Graph $K_{n,n}$). *Let $\mathcal{H} = \{\{i, n+j\} \mid i, j = 1, \ldots, n\}$, where n is a positive integer. Its dual hypergraph is $\mathcal{H}^d = \{\{1, 2, \ldots, n\}, \{n+1, n+2, \ldots, 2n\}\}$.*

For any subset X of V, let $\bar{X}$ denotes the compliment of X, i. e., $\bar{X} \overset{\text{def}}{=} V \setminus X$. Moreover, let $\mathcal{H}^c$ denote the hypergraph consisting of compliments of hyperedges in $\mathcal{H}$, i. e., $\mathcal{H}^c \overset{\text{def}}{=} \{\bar{H} | H \in \mathcal{H}\}$.

A subset of vertices $I \in V$ is called an *independent set* of the hypergraph $\mathcal{H} \subseteq 2^V$ when I does not contain any hyperedge of $\mathcal{H}$, i. e., $H \not\subseteq I$, for all $H \in \mathcal{H}$. It is *maximal* when any of its proper supersets is not an independent set. Note that if I is an independent set of $\mathcal{H}$ then $\bar{I} \cap H \neq \emptyset$ for all $H \in \mathcal{H}$, i. e., $\bar{I}$ is a transversal of $\mathcal{H}$. In particular, $\bar{I}$ is a minimal transversal of $\mathcal{H}$ if and only if I is a maximal independent set of $\mathcal{H}$. Consequently, the set of all maximal independent sets is equal to the hypergraph $\mathcal{H}^{dc} \overset{\text{def}}{=} (\mathcal{H}^d)^c$.

For a hypergraph $(V, \mathcal{H} = \{H_1, \ldots, H_m\})$, let us define its *transposed* hypergraph $\mathcal{H}^t$ whose vertices corresponds to hyperedges of $\mathcal{H}$ and every vertex $u \in V$ defines a hyperedge $\{H \mid H \ni u\}$ in $\mathcal{H}^t$.

A hyperedge $H \in \mathcal{H}$ *covers* a vertex $u \in V$ when $H \ni u$. A subset of hyperedges $\mathcal{H}' \subseteq \mathcal{H}$ is called a *covering* of hypergraph $(V, \mathcal{H})$ when every vertex is covered by some hyperedge in $\mathcal{H}'$, i. e., $\bigcup_{H \in \mathcal{H}'} H = V$. Note that by definition, $\mathcal{H}' \subseteq \mathcal{H}$ is a covering of V if and only if $\mathcal{H}'$ is a transversal of the transposed hypergraph $\mathcal{H}^t$ and so the problem of finding all minimal coverings of $\mathcal{H}$ is equivalent to the problem of finding all minimal transversals of its transposed hypergraph $\mathcal{H}^t$.

For any hypergraph $\mathcal{H} \subseteq 2^V$ and a subset $S \subseteq V$, we use the following notations: $\mathcal{H}_S$ denotes the sub-hypergraph induced by the vertices in S, i. e., $\mathcal{H}_S \overset{\text{def}}{=} \{H \in \mathcal{H} \mid H \subseteq S\}$, and $\mathcal{H}^S$ denotes the projection of $\mathcal{H}$ on S, i. e., $\mathcal{H}^S \overset{\text{def}}{=} \min(\{H \cap S \mid H \in \mathcal{H}\})$. Note that $\mathcal{H}^S$ is empty when S is not a hitting set of $\mathcal{H}$.

Given two hypergraphs $\mathcal{H}_1$ and $\mathcal{H}_2$ with vertex set V, denote by

$$\mathcal{H}_1 \wedge \mathcal{H}_2 \overset{\text{def}}{=} \min\{H_1 \cup H_2 \,|\, H_1 \in \mathcal{H}_1 \text{ and } H_2 \in \mathcal{H}_2\},$$
$$\mathcal{H}_1 \vee \mathcal{H}_2 \overset{\text{def}}{=} \min(\mathcal{H}_1 \cup \mathcal{H}_2)$$

the conjunction and disjunction of $\mathcal{H}_1$ and $\mathcal{H}_2$, respectively.

Let us denote the *degree* of a vertex in hypergraph $\mathcal{H}$ by $\deg_{\mathcal{H}}(v)$, which is the number of hyperedges of $\mathcal{H}$ containing $v \in V$. Also, we sometimes write $V \setminus v$ instead of $V \setminus \{v\}$ for brevity.

A hypergraph $\mathcal{H}$ is said to be *k-Helly* if for any $\mathcal{H}' \subseteq \mathcal{H}$ the following holds: if every k hyperedges in $\mathcal{H}'$ have a non-empty intersection then all edges in $\mathcal{H}'$ have a non-empty intersection.

A hypergraph is said to be *k-conformal* [Ber89] if any set $X \subseteq V$ is contained in a hyperedge of $\mathcal{H}$ whenever each subset of X of cardinality at most k is contained in a hyperedge of $\mathcal{H}$. We have the following proposition.

Proposition 2.3 (cf. [Ber89]). *A hypergraph $\mathcal{H}$ is k-Helly if and only if its transpose $\mathcal{H}^t$ is k-conformal.*

A hypergraph $\mathcal{H}$ is said to be *k-degenerate* [EGM02] if for every set $X \subseteq V = V(\mathcal{H})$, the minimum degree of a vertex in the induced hypergraph $\mathcal{H}_X$ on X is most k. Let $v_1 \in V$ be a vertex with minimum degree in $\mathcal{H}$. Since $\mathcal{H}$ is k-degenerate, we know that $\deg_{\mathcal{H}}(v) \leq k$. Now consider the induced hypergraph $\mathcal{H}_{V \setminus v_1}$ on the remaining vertices. By the definition of k-degeneracy, there exists a vertex v_2 in $V \setminus v_1$ such that degree of v_2 in $\mathcal{H}_{V \setminus v_1}$ is at most k. In particular a vertex with minimum degree in the induced hypergraph satisfies this property. Consequently, we get an ordering $v_1, \ldots, v_n$ of vertex set V such that for each $1 \leq i \leq n$, the degree of v_i in $\mathcal{H}_{V \setminus \{v_i, \ldots, v_n\}}$ is at most k. In fact, it can be easily shown that a hypergraph is k-degenerate if and only if there exists such ordering of its vertices. Note that the hypergraphs in which every vertex has degree at most k are also k-degenerate.

A hypergraph $\mathcal{H}$ is said to be *r-exact* if any minimal transversal of $\mathcal{H}$ intersects any hyperedge of $\mathcal{H}$ in at most r vertices.

2.2 Properties of the Transversal Hypergraph

The following lemma gives a necessary and sufficient condition for one hypergraph to be dual of another. See, for example, [Ber89, Chapter 2, Page 44] for details.

Lemma 2.4 (Vertex-coloring lemma). *Let $\mathcal{G}, \mathcal{H}$ be two Sperner hypergraphs on a set V. Then $\mathcal{H} = \mathcal{G}^d$ if and only if every pair (A, B) with $A, B \subset V, A \cup B = V, A \cap B = \emptyset$, satisfies:*
(i) there exists either an $H \in \mathcal{H}$ contained in A or a $G \in \mathcal{G}$ contained in B;
(ii) these two cases cannot happen simultaneously.

Corollary 2.5. *Let $\mathcal{G}, \mathcal{H}$ be two Sperner hypergraphs on a set V. Then $\mathcal{H} = \mathcal{G}^d$ if and only if $\mathcal{G} = \mathcal{H}^d$.*

Proof. Indeed $\mathcal{H} = \mathcal{G}^d$ if and only if every pair (A, B) satisfies (i) and (ii) with $\mathcal{G}, \mathcal{H}$; that is every pair (B, A) satisfies (i) and (ii) with $\mathcal{H}, \mathcal{G}$; that is $\mathcal{G} = \mathcal{H}^d$. □

The following is a straightforward consequence of the above corollary.

Corollary 2.6. *Let $\mathcal{H}$ be a Sperner hypergraph. Then $(\mathcal{H}^d)^d = \mathcal{H}$.*

The following proposition presents some necessary conditions for the pair of hypergraphs to be dual of each other.

Proposition 2.7 ([FK96]). *Let $\mathcal{G}, \mathcal{H}$ be two Sperner hypergraphs on a set V. If $\mathcal{H} = \mathcal{G}^d$ then*

$$G \cap H \neq \emptyset \textit{ for all } G \in \mathcal{G}, H \in \mathcal{H}, \tag{2.1}$$

$$\cup\{G : G \in \mathcal{G}\} = \cup\{H : H \in \mathcal{H}\}, \tag{2.2}$$

$$\max\{|G| : G \in \mathcal{G}\} \leq |\mathcal{H}| \textit{ and } \max\{|H| : H \in \mathcal{H}\} \leq |\mathcal{G}|. \tag{2.3}$$

Proof. (i) Suppose Equation (2.1) does not hold for some $G \in \mathcal{G}, H \in \mathcal{H}$. Then the pair $(H, V \setminus H)$ violates the condition (ii) of Lemma 2.4.
(ii) Let $i \in G$ for some $G \in \mathcal{G}$ and $i \notin H$ for all $H \in \mathcal{H}$. We show that the pair $(G \setminus \{i\}, (V \setminus G) \cup \{i\})$ violates the condition (i) of Lemma 2.4. Indeed $G \setminus \{i\}$ does not contain any hyperedge of Sperner hypergraph $\mathcal{G}$. Also, since every $H \in \mathcal{H}$ hits $G \setminus \{i\}$, the set $(V \setminus G) \cup \{i\}$ does not contain any hyperedge of $\mathcal{H}$. The other case when there is $j \in H$ for some $H \in \mathcal{H}$ and $j \notin G$ for all $G \in \mathcal{G}$ is completely symmetric.
(iii) Let $G \in \mathcal{G}$ such that $|G| > |\mathcal{H}|$ then clearly G is a not minimal transversal of $\mathcal{H}$ by Pigeonhole principle. The other case when there is $H \in \mathcal{H}$ such that $|H| > |\mathcal{G}|$ is symmetric. □

In the following we discuss properties of a single minimal transversal and its subsets. Let $\mathcal{G}$ be a hypergraph on a set V. Observe that for every minimal transversal T of the hypergraph $\mathcal{G}$ and for any $v \in T$, there is always some edge $G \in \mathcal{G}$ which *requires* v, i. e., $G \cap T = \{v\}$. We call such a edge a *certificate* edge for v in T.

A subset of vertices $S \subseteq V$ is called a *sub-transversal* of $\mathcal{G}$ if it is contained in some minimal transversal T of $\mathcal{G}$, i. e., $S \subseteq T$ and $T \in \mathcal{G}^d$. Given a hypergraph $\mathcal{G} \subseteq 2^V$ and a subset $S \subseteq V$ of vertices, [BGH98] gave a criterion to decide if S is a sub-transversal of $\mathcal{G}$.

To describe the sub-transversal criterion, we need a few more definitions. For a subset $S \subseteq V$, and a vertex $v \in S$, let $\mathcal{G}_v(S) = \{G \in \mathcal{G} \mid G \cap S = \{v\}\}$. A selection of $|S|$ hyperedges $\{G_v \in \mathcal{G}_v(S) \mid v \in S\}$ is called *blocked* if there exists a hyperedge $G \in \mathcal{G}_0 \overset{\text{def}}{=} \{G \in \mathcal{G} \ : \ G \cap S = \emptyset\}$ such that $G \subseteq \bigcup_{v \in S} G_v$.

Proposition 2.8 ([BGH98]). *A non-empty subset $S \subseteq V$ is a sub-transversal for $\mathcal{G} \subseteq 2^V$ if and only if there is a non-blocked selection $\{G_v \in \mathcal{G}_v(S) \mid v \in S\}$ for S.*

It is not hard to see that the condition in Proposition 2.8 is necessary, because we can think of a selection as a set of tentative certificates for vertices in S, and for any blocking selection, at least one of the certificate clearly becomes invalid when S is extended to hit the edge which blocks the selection under consideration. So there must exists a non-blocked selection to be able to extend S to a minimal transversal.

2.3 Hypergraph Transversal Problem

Given a hypergraph as input, the *hypergraph transversal problem* asks to generate its dual. Formally, we define the problem as follows.

Problem DUALIZATION

Input: Sperner hypergraph $\mathcal{G}$.

Output: The dual hypergraph $\mathcal{G}^d$.

As Example 2.1 shows, the size of the dual hypergraph can be exponential in the size of the input hypergraph. So typically enumeration problems are analyzed in terms of both input and output sizes. More concretely, let n be the number of vertices, m be the number of hyperedges in the input hypergraph and m' be the number of edges in the dual hypergraph. We say an algorithm for DUALIZATION is *output polynomial* when its running time can be bounded by some polynomial in n, m and m'. Moreover, an algorithm for DUALIZATION is *incremental polynomial* when it enumerates one by one all minimal transversals of the input hypergraph such that the time it takes to output another minimal transversal is polynomial in n, m and k, the last parameter being the size of dual hypergraph generated so far.

The notion of incremental polynomial algorithm leads to the following problem of deciding whether a given partial list of minimal transversals is complete for the given hypergraph. Formally, we define the problem as follows.

Problem INCRDUAL

Input: Sperner hypergraph $\mathcal{F}$ and a subset of its minimal transversals $\mathcal{G} \subseteq \mathcal{F}^d$.

Output: True if $\mathcal{G} = \mathcal{F}^d$, otherwise return a new transversal from $\mathcal{F}^d \setminus \mathcal{G}$.

Finally, we define the problem of deciding whether the given two hypergraphs are dual to each other.

Problem DUAL

Input: Sperner hypergraphs $\mathcal{F}$ and $\mathcal{G}$.

Output: True if $\mathcal{G} = \mathcal{F}^d$, False otherwise.

It is easy to see that by making $m' + 1$ calls to an algorithm for INCRDUAL, we can generate all minimal transversals and hence a polynomial time algorithm for INCRDUAL implies an incremental polynomial algorithm for DUALIZATION. Bioch and Ibaraki [BI95] show the following stronger result.

Proposition 2.9. *The existence of any of the the following algorithms implies all of the others.*

1. *An incremental polynomial time algorithm for* DUALIZATION.
2. *A polynomial time algorithm for* INCRDUAL.
3. *A polynomial time algorithm for* DUAL.

An output polynomial algorithm may be too expensive for some applications and so there is a notion of a *polynomial delay* algorithm in which the time to produce each successive minimal transversal is polynomial in the size of input hypergraph only. Therefore, the total running time of a polynomial delay algorithm to generate all minimal transversals would be $\mathrm{poly}(n, m) \cdot m'$.

Clearly, the problem $\textsc{Dual}(\mathcal{G}, \mathcal{H})$ is in co-NP since if $\mathcal{H}$ is not the transversal hypergraph of $\mathcal{G}$ then this can be witnessed by a set $X \subseteq V$ such that

$$X \cap G \neq \emptyset \text{ for all } G \in \mathcal{G}, \text{ and } X \not\supseteq H \text{ for all } H \in \mathcal{H}. \tag{2.4}$$

Intuitively, X is a transversal of $\mathcal{G}$ (not necessarily minimal) that does not include any hyperedge of $\mathcal{H}$. Such a set X is a *witness* for the non-duality of the pair $(\mathcal{G}, \mathcal{H})$. Note that the condition (2.4) is symmetric in $\mathcal{G}$ and $\mathcal{H}$: $X \subseteq V$ satisfies (2.4) for the pair $(\mathcal{G}, \mathcal{H})$ if and only if $V \setminus X$ satisfies (2.4) for $(\mathcal{H}, \mathcal{G})$. Also, by definition, the pair $(\emptyset, \{\emptyset\})$ is dual.

2.4 Enumeration Techniques

Is this section, we review some techniques from the literature to solve the hypergraph transversal problem. We also present simple sub-routines in Section 2.4.1.1 which will be used to solve small instances confronted later as base-cases in more sophisticated algorithms.

2.4.1 Berge Multiplication

The following proposition is elementary and follows from the basic definitions.

Proposition 2.10. *Given hypergraphs $\mathcal{H}_1, \ldots, \mathcal{H}_k \subseteq 2^V$,*

$$(\mathcal{H}_1 \vee \cdots \vee \mathcal{H}_k)^d = \mathcal{H}_1^d \wedge \cdots \wedge \mathcal{H}_k^d.$$

Proof. Let T be a minimal transversal of $\mathcal{H} = \bigvee_{i \in [k]} \mathcal{H}_i$ and let $T_i = T \cap V(\mathcal{H}_i)$ for $i \in [k]$. Clearly T_i is a transversal of $\mathcal{H}_i$ which is not necessarily minimal. Let T_i' be a minimal transversal of $\mathcal{H}_i$ contained in T_i and let $T' = \bigcup_{i \in [k]} T_i'$. We show that $T' = T$ and so $T \in \bigwedge_{i \in k} \mathcal{H}_i^d$. Indeed, the existence of a vertex $v \in T \setminus T'$ contradicts the assumption that T is a minimal transversal of $\mathcal{H}$.

To see the other direction, let T be a set in $\bigwedge_{i \in k} \mathcal{H}_i^d$. By definition $T = \bigcup_{i \in [k]} T_i$, where $T_i \in \mathcal{H}_i^d$ for $i \in [k]$ and there is no $T' \subset T$ with $T' = \bigcup_{i \in [k]} T_i'$, where $T_i' \in \mathcal{H}_i^d$ for $i \in [k]$. This implies that T is a minimal transversal of $\mathcal{H} = \bigvee_{i \in [k]} \mathcal{H}_i$ since the existence of smaller transversal $T'' \subset T$ of $\mathcal{H}$ would yield a smaller transversal of $\mathcal{H}_i$ for some $i \in [k]$. □

Based on Proposition 2.10 we can find all minimal transversals of a given hypergraph by processing its edges one by one and computing all of the minimal transversal of the partial hypergraph read so far. The algorithm which is attributed to Berge [Ber89] works similar to the multiplication of algebraic expressions and hence is sometimes called *Berge Multiplication* in the literature.

Algorithm 1 Berge Multiplication

```
Input: A hypergraph $\mathcal{H} = \{H_1, \ldots, H_m\}$
Output: The dual hypergraph $\mathcal{H}^d$
1: $\mathcal{X} := \{\emptyset\}$
2: for $i = 1, \ldots, m$ do
3:     $\mathcal{X} := \mathcal{X} \wedge \{\{u\} \mid u \in H_i\}$
4: return $\mathcal{X}$
```

The correctness of Algorithm 1 follows from Proposition 2.10 and the the invariant that $\mathcal{X}$ is the transversal hypergraph of $\{H_1, \ldots, H_i\}$ after every ith iteration for $i = 1, \ldots, m$ and hence the hypergraph returned by the procedure is indeed the transversal hypergraph of the input hypergraph $\mathcal{H}$. To analyze the running time, let Γ be the maximum size of the intermediate hypergraph $\mathcal{X}$ in Step 3 of Algorithm 1. Then its running time can be bounded by $O(mn\Gamma \min\{m, \Gamma\})$ [BEM10].

The drawback of this otherwise simple approach is that it does not in general yield an output polynomial algorithm. To see this, consider the hypergraph in Example 2.2 given as input to Algorithm 1. If the algorithm first multiplies the n edges of the form $\{\{i, n+i\} \mid i = 1, \ldots, n\}$, then the size of intermediate hypergraph $\mathcal{X}$ in Algorithm 1 is exponential in the input plus output sizes, whereas there are only two distinct minimal transversals of the complete hypergraph.

The above example illustrates that Berge multiplication is sensitive to the order in which it processes hyperedges of the input hypergraph. This raises the natural question whether there always exists an ordering in which it is output polynomial. The answer turns out to be No as shown by Takata [Tak07] which gives a family of hypergraphs for which Berge multiplication is not output polynomial for every possible ordering of their hyperedges. On the other hand, sub-exponential bounds are known [BEM08, BEM10].

2.4.1.1 Dualizing Small Instances

In this thesis, we use Algorithm 1 to solve small instances of DUALIZATION($\mathcal{G}$) and DUAL($\mathcal{G}, \mathcal{H}$) respectively, which arise as base-cases in more sophisticated algorithms later in the thesis.

The sub-routine **Dualize-Simple**($\mathcal{G}$) uses multiplication (Algorithm 1) to generate $\mathcal{G}^d$. While, for an input pair of hypergraphs $\mathcal{G}, \mathcal{H} \subseteq 2^V$, the sub-routine **Dual-Simple**($\mathcal{G}, \mathcal{H}$) uses multiplication to get $\mathcal{G}^d$ and compares it with $\mathcal{H}$ to test whether $\mathcal{G}^d = \mathcal{H}$ or not. Note that for hypergraphs with constant number of edges c, both sub-routines run in $O(n^c)$ time, where n is the number of vertices in V.

2.4.2 Backtracking Method

Let $\mathcal{H}$ be a hypergraph on a set V. Recall that a subset of vertices $S \subseteq V$ is called a *sub-transversal* of $\mathcal{H}$ if it is contained in some minimal transversal T of $\mathcal{H}$, i.e., $S \subseteq T$ and $T \in \mathcal{H}^d$. Given an oracle to decide whether a given set is sub-transversal or not, we can generate the transversal hypergraph as shown in Algorithm 2. The procedure is based on the standard backtracking technique for enumeration (see e.g. [RT75, Eit94]) and is called initially with $S = \emptyset$. It is easy to verify that the algorithm outputs all elements of the transversal hypergraph $\mathcal{H}^d$, without repetition. Since the algorithm essentially builds a backtracking tree whose leaves are the minimal transversals of $\mathcal{G}$, the time required to produce each new minimal transversal is bounded by the depth of the tree

Algorithm 2 The backtracking method for finding minimal transversals

```
Procedure Dualize-BT($\mathcal{H}, S, V$):
Input: A hypergraph $\mathcal{H} \subseteq 2^V$, and a subset $S \subseteq V$
Output: The set $\{T \in \mathcal{H}^d \ : \ T \supseteq S\}$
1: if $S \in \mathcal{H}^d$ then
2:     output $S$ and return
3: if $\exists e \in V \setminus S$, such that $S \cup \{e\}$ is a sub-transversal for $\mathcal{H}$ then
4:     Dualize-BT($\mathcal{H}, S \cup \{e\}, V$)
5:     Dualize-BT($\mathcal{H}^{V \setminus \{e\}}, S, V \setminus \{e\}$)
```

(at most $|V|$) times the maximum time required at each node. Assuming the the test in Step 3 takes time Γ_1, we get the following running time of Algorithm 2.

Lemma 2.11. *Given a hypergraph $\mathcal{H}$ on a set V, Algorithm 2 generates $\mathcal{H}^d$ with delay $O(n^2\Gamma_1)$, where $n = |V|, m = |\mathcal{H}|$ and Γ_1 is the upper bound on the time taken by the the sub-transversal test in Line 3.*

2.4.3 Incremental Method

The following proposition is a generalization of a similar observation made for graphs in [LLK80] (see also [EGM03]).

Proposition 2.12. *Let $\mathcal{H}$ be a hypergraph on a set V. Then $(\mathcal{H}_S)^d = (\mathcal{H}^d)^S$.*

Proof. The claim trivially holds when the hypergraph $\mathcal{H}' = \min(\mathcal{H}) \setminus \mathcal{H}_S$ is empty, so for the rest of the proof we assume otherwise.

Let T_s be a minimal transversal of $\mathcal{H}_S$. Clearly, $T \cap S \not\subset T_s$ for any $T \in \mathcal{H}^d$ because of minimality of T_s. Also since $\min(\mathcal{H})$ is Sperner, there must exist a $T \in \mathcal{H}^d$ such that T hits edges in $\mathcal{H}'$ with vertices from $V \setminus S$ and $T \cap S = T_s$. The above two facts imply that $T_s \in (\mathcal{H}^d)^S$.

To prove the other direction, let $T_s \in (\mathcal{H}^d)^S$. Clearly T_s is a transversal of $\mathcal{H}_S$. We need to prove that it is also a minimal transversal. Note that by definition, $(\mathcal{H}^d)^S = \min\{T \cap S \ : \ T \in \mathcal{H}^d\}$. Let T be a minimal transversal of $\mathcal{H}$ that realizes T_s, i. e., $T \cap S = T_s$. Clearly, if T_s is not minimal then T is also not minimal, which is a contradiction to our assumption that $T \in \mathcal{H}^d$. □

Combining the above with the fact that $(\mathcal{H}^d)^d = \mathcal{H}$ (Corollary 2.6), we get the following.

Corollary 2.13. *Let $\mathcal{H}$ be a hypergraph on a set V. Then $(\mathcal{H}^S)^d = (\mathcal{H}^d)_S$.*

For an ordering of vertices $V = \{v_1, v_2, \ldots, v_n\}$, let $\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_n$ be a partition of hypergraph $\mathcal{H}$ defined as $\mathcal{H}_i = \{H \in \mathcal{H} : H \ni v_i, H \subseteq \{v_1, \ldots, v_i\}\}$. Then, the following corollary directly follows from Proposition 2.12.

Corollary 2.14. *For all i: $|(\mathcal{H}_1 \cup \ldots \cup \mathcal{H}_i)^d| \leq |\mathcal{H}^d|$ and for every $X \in (\mathcal{H}_1 \cup \ldots \cup \mathcal{H}_{i-1})^d$,*

$$|(\{H \in \mathcal{H}_i : H \cap X = \emptyset\})^d| \leq |(\mathcal{H}_1 \cup \ldots \cup \mathcal{H}_i)^d|.$$

Proof. Both follow from Proposition 2.12 with S being $\{v_1, \ldots, v_i\}$ or $\{v_1, \ldots, v_i\} \setminus X$ respectively. □

Algorithm 3 Sequential method for finding minimal transversals

```
Dualize-Inc($\mathcal{H}, V = \{v_1, \ldots, v_n\}$):
Input: A hypergraph $\mathcal{H} \subseteq 2^V$ and an ordering of vertices
Output: The set $\mathcal{H}^d$
1: $\mathcal{X}_0 \leftarrow \{\emptyset\}$ and for all $1 \leq i \leq n : \mathcal{X}_i \leftarrow \{\}$
2: for $i = 1, \ldots, n$ do
3:     for all $X \in \mathcal{X}_{i-1}$ do
4:         $\mathcal{Y} \leftarrow$ Dualize-Sub($\{H \in \mathcal{H}_i : H \cap X = \emptyset\}$)
5:         if $X \cup Y \in (\mathcal{H}_1 \cup \ldots \cup \mathcal{H}_i)^d$ for any $Y \in \mathcal{Y}$ then
6:             $\mathcal{X}_i \leftarrow \mathcal{X}_i \bigcup \{X \cup Y\}$
7: return $\mathcal{X}_n$
```

By exploiting the above properties, Algorithm 3 generates the transversal hypergraph of $\mathcal{H}$ as follows: it proceeds inductively, for $i = 1, \ldots, n$, by finding $(\mathcal{H}_1 \cup \ldots \cup \mathcal{H}_{i-1})^d$. Then for each set X in this transversal hypergraph it extends X to a minimal transversal of $(\mathcal{H}_1 \cup \ldots \cup \mathcal{H}_i)^d$ by finding $(\{H \in \mathcal{H}_i : H \cap X = \emptyset\})^d$, each set of which is combined with X, to obtain a candidate for a minimal transversal of $\mathcal{H}_1 \cup \ldots \cup \mathcal{H}_i$. Note that in Line 4 we call a subroutine **Dualize-Sub**(). As we will see in the later chapters, Algorithm 3 leads to number of efficient algorithms when this step can be performed appropriately. Assuming the correctness of the subroutine **Dualize-Sub()**, the correctness of Algorithm 3 follows from Proposition 2.10.

Let Γ_2 be the upper bound on the running time of the subroutine **Dualize-Sub**. To analyze the running time of Algorithm 3, consider the i-th iteration. From Corollary 2.14 we have $|\mathcal{X}_{i-1}| \leq |\mathcal{H}^d|$ and so there are at most $|\mathcal{H}^d|$ calls to **Dualize-Sub**. The size of $\mathcal{Y}$ in step 4 can also be bounded by Corollary 2.14, which gives us $|\mathcal{Y}| \leq |\mathcal{H}^d|$. Finally the condition in step 5 can be checked in time $O(n|\mathcal{H}|)$. Thus the time spent in the i-th iteration can be bounded by $O\left(|\mathcal{H}^d|\left(\Gamma_2 + n|\mathcal{H}||\mathcal{H}^d|\right)\right)$.

Lemma 2.15. *Given a hypergraph $\mathcal{H}$ on a set V, Algorithm 3 generates $\mathcal{H}^d$ in time $O(nm'(\Gamma_2 + nmm'))$, where $n = |V|, m = |\mathcal{H}|, m' = |\mathcal{H}^d|$ and Γ_2 is the upper bound on the time taken by the sub-routine* **Dualize-Sub***() in step 4.*

2.4.4 Divide and Conquer

The following decomposition rule which is due to Fredman and Khachyian [FK96] divides the problem into two subproblems not containing a given vertex $v \in V$.

Proposition 2.16 (cf. [FK96]). *Let $\mathcal{G}, \mathcal{H} \subseteq 2^V$ be two hypergraphs satisfying* (2.1), *and $v \in V$ be a given vertex. Then $\mathcal{G}$ and $\mathcal{H}$ are dual if and only if the pairs $(\mathcal{G}_{V\setminus v}, \mathcal{H}^{V\setminus v})$ and $(\mathcal{G}^{V\setminus v}, \mathcal{H}_{V\setminus v})$ are dual.*

The above proposition gives a divide and conquer approach for the hypergraph dualization problem. Fredman and Khachyian [FK96] also show that for a pair of hypergraphs $\mathcal{G}, \mathcal{H} \subseteq 2^V$ satisfying equations (2.1), (2.2) and (2.3), there exist a vertex $v \in V$ contained in at least $1/\log(|\mathcal{G}| + |\mathcal{H}|)$ fractions of the edges in either $\mathcal{G}$ or $\mathcal{H}$. Using such high frequency vertex for decomposition in Proposition 2.16 yields the following theorem.

Theorem 2.17 ([FK96]). *Let $\mathcal{G}, \mathcal{H} \subseteq 2^V$ be a pair of hypergraph satisfying* (2.1), (2.2) *and* (2.3) *and let $N = |\mathcal{G}| + |\mathcal{H}|$. Then* $\text{DUAL}(\mathcal{G}, \mathcal{H})$ *can be decided in $N^{O(\log^2(N))}$ time.*

The above result can be further improved by observing that the two subproblems in Proposition 2.16 are not independent. By further decomposing one of them gives the following theorem.

Theorem 2.18 ([FK96]). *Let $\mathcal{G}, \mathcal{H} \subseteq 2^V$ be a pair of hypergraph satisfying* (2.1), (2.2) *and* (2.3) *and let $N = |\mathcal{G}| + |\mathcal{H}|$. Then* $\textsc{Dual}(\mathcal{G}, \mathcal{H})$ *can be decided in $N^{o(\log(N))}$ time.*

Note that the above theorem gives the best known upper bound on the complexity of hypergraph dualization problem. For alternate algorithms with similar upper bounds, see [Tam00, GK04, Elb06, BM09].

2.4.5 Full Cover Decompositions

We next describe another decomposition. Call a family of sets $\{S_1, \ldots, S_r\} \subseteq 2^V$ a *full cover* of $\mathcal{H}$ if for every $H \in \mathcal{H}$ there is an $i \in [r]$ such that $H \subseteq S_i$. In other words.

$$\mathcal{H} = \bigcup_{i \in [r]} \mathcal{H}_{S_i}$$

For example, $\{V\}$ and $\mathcal{H}$ are clearly full covers of $\mathcal{H}$. The next lemma states that a full cover of $\mathcal{H}$ can be used to decompose the hypergraph transversal problem into k subproblems.

Lemma 2.19. *Let $\mathcal{S} = \{S_1, \ldots, S_r\} \subseteq 2^V$ be a full cover of a hypergraph $\mathcal{H} \subseteq V$. Then*

$$\mathcal{H}^d = \bigwedge_{i \in r} \mathcal{H}^d_{S_i}.$$

The above decomposition was initially suggested in [KBEG07a, KBGE07], and used in a number of subsequent works [Elb08, BM09]. So far, the use of such decompositions has been successful for developing polynomial time dualization algorithms for limited cases, such as hypergraphs of bounded size or bounded degree. In Chapter 5, we shall extend this technique to geometric hypergraphs.

The decomposition in Lemma 2.19 immediately suggests a parallel algorithm for the problem which is shown as Algorithm 4. For $r, s_0 \in \mathbb{Z}_+$ and $0 < \epsilon < 1$, denote by $\mathbb{H}(r, \epsilon, s_0)$ the family of hypergraphs $\mathcal{H} \subseteq 2^V$, such that for every $S \subseteq V$ with $|S| \geq s_0$, there exist subsets $S_1, \ldots, S_r \subseteq S$ satisfying:

(H1) $S_1, \ldots, S_r \subseteq S$ forms a full cover of $\mathcal{H}_S$;

(H2) $|S_i| \leq (1 - \epsilon)|S|$, for each $i \in \{1, \ldots, r\}$.

Algorithm 4 Dualizing hypergraphs satifying (H1) and (H2).

```
**Dualize**($\mathcal{H}, V$):
**Input:** A hypergraph $\mathcal{H} \in \mathbb{H}(r, \epsilon, s_0)$
**Output:** The set $\mathcal{H}^d$
1: If $|\mathcal{H}| \in \{0, 1\}$ or $|V| \leq s_0$, then return $\bigwedge_{H \in \mathcal{H}} H^d$
2: In parallel, do the following:
3:     Find the sets $S_1, \ldots, S_r \subseteq V$ satisfying (H1) and (H2) with $S = V$
4:     Let $\mathcal{G}_i \leftarrow$ **Dualize**($\mathcal{H}_{S_i}, S_i$), for $i = 1, \ldots, r$
5:     Compute the conjunction $\mathcal{G} \leftarrow \wedge_{i=1}^{r} \mathcal{G}_i$
6: **return** $\mathcal{G}$
```

Khachyian et al. [KBEG07a] showed that given a hypergraph from $\mathbb{H}(r, \epsilon, s_0)$ as input, Algorithm 4 output all minimal transversals in parallel in time polylogarithmic in the input and output sizes (in the PRAM model). Let $\tau = \tau(n, |\mathcal{H}|, r, \epsilon)$ be the time and $\pi = \pi(n, |\mathcal{H}|, r, \epsilon)$ be the number of processors to compute a full cover of the input hypergraph satisfying (H1) and (H2).

Lemma 2.20. *Let $t(n, m')$ and $p(n, m')$ be respectively the time and the number of processors, required by Algorithm 4 to output all minimal transversals of a hypergraph $\mathcal{H} \in \mathbb{H}(r, \epsilon, s_0)$ on n vertices and with $|\mathcal{H}^d| = m'$. Then $t(n, m') = O((\tau + \log n + r \log m')\frac{\log n}{\epsilon})$ and $p(n, m') = O((\pi + m'^{2r}) \cdot n^{(\log r)/\epsilon + 2})$*

Note that Algorithm 4 is output polynomial and that all minimal transversals are generated simultaneously at the very end. Using techniques from [KBEG07a], we can also get an incremental version of this, i.e., for any $k \leq m'$, the running time depends polylogarithmically on k, provided that there is an efficient parallel algorithm for finding a single minimal transversal of the input hypergraph $\mathcal{H}$. The existence of the latter algorithm for general hypergraphs, is an outstanding open question (see e.g. [KUW88]). The currently best known parallel implementation for the later problem is due to Karp, Upfal, and Wigderson [KUW88] who gave a randomized algorithm which makes only $O(\sqrt{n})$ parallel oracle calls on $O(n^{3/2})$ processors to compute a maximal independent set (complement of a minimal transversal, in the case of explicitly given hypergraphs) of an independence system given by an oracle on n vertices.

The incremental algorithm is presented as Algorithm 5. The technique works by ensuring at every decomposition step that no more than $O(k)$ partial minimal transversals are generated. If at some point in the algorithm, k minimal transversals are already generated then we complete them and stop the procedure.

Lemma 2.21. *Algorithm 5 outputs k (or all if $|\mathcal{H}^d| \leq k$) minimal transversals of a hypergraph $\mathcal{H} \in \mathbb{H}(r, \epsilon, s_0)$ on n vertices in time $t(n, m') = O((\tau + \log n + r \log k)\frac{\log n}{\epsilon} + \Delta)$ and $p(n, m') = O((\pi + k^{2r}) \cdot n^{(\log r)/\epsilon+2} + k\Pi)$ processors, where Δ and Π are, respectively, the parallel time and the number of processors required to generate a single minimal transversal of $\mathcal{H}$.*

Algorithm 5 Dualizing $\mathcal{H} \in \mathbb{H}(r, \epsilon, s_0)$ incrementally and in parallel.

```
Dualize($\mathcal{H}, V$):
Input: A hypergraph $\mathcal{H} \in \mathbb{H}(r, \epsilon, s_0)$
Output: The set $\mathcal{H}^d$
 1: If $|\mathcal{H}| \in \{0, 1\}$ or $|V| \leq s_0$, then return $\bigwedge_{H \in \mathcal{H}} H^d$
 2: In parallel, do the following:
 3:    Find the sets $S_1, \ldots, S_r \subseteq V$ satisfying (H1) and (H2) with $S = V$
 4:    Let $\mathcal{G}_i \leftarrow$ Dualize($\mathcal{H}_{S_i}, S_i$), for $i = 1, \ldots, r$
 5:    If there is an $i \in \{1, \ldots, r\}$ such that $|\mathcal{G}| = k$ then
 6:       In parallel, for each $Y \in \mathcal{G}_i$, do the following:
 7:          Let $\mathcal{H}[Y] = \{H \setminus S_i | H \cap Y = \emptyset\}$
 8:          Compute a minimal transversal $T_Y$ of $\mathcal{H}[Y]$
 9:          Return $\mathcal{G} \leftarrow \{Y \cup T_Y | Y \in \mathcal{G}_i\}$, and stop
10:    Else compute the conjunction $\mathcal{G} \leftarrow \wedge_{i=1}^{r} \mathcal{G}_i$
11: return $\mathcal{G}$ (truncated to $k$ elements if $|\mathcal{G}| > k$)
```

In this thesis, we extend the results of [KBEG07a] and show that even with either condition (H1) or (H2) relaxed, we can still find incremental polynomial or efficient parallel algorithms for some classes of hypergraphs as we will see later in Chapter 4 and Chapter 5.

2.5 Related Problems

2.5.1 Dualization of Monotone Boolean Functions

For vectors $x = \{x_1, \ldots, x_n\}$ and $x' = \{x'_1, \ldots, x'_n\}$ in $\{0,1\}^n$, we say $x \leq x'$ when each component of x is less then or equal to the corresponding component of x' i. e., $x_i \leq x'_i$ for $i = 1, \ldots, n$. A Boolean function $f : \{0,1\}^n \to \{0,1\}$ is called *monotone* whenever for any $x, x' \in \{0,1\}^n,\ x \leq x'$ implies $f(x) \leq f(x')$.

A *minterm* (*maxterm*) of a monotone Boolean function is a minimal set of variables which, if all assigned the value 1 (resp., value 0), forces the function to take the value 1 (resp., value 0) regardless of the values assigned to the remaining variables. Any monotone Boolean function f has unique irredundant (i. e., when no term contains another) *disjunctive normal form* (DNF) and *conjunctive normal form* (CNF)

$$\varphi = \bigvee_{F \in \mathcal{F}} \Big(\bigwedge_{i \in F} x_i \Big) \tag{2.5}$$

$$\psi = \bigwedge_{G \in \mathcal{G}} \Big(\bigvee_{i \in G} x_i \Big), \tag{2.6}$$

where $\mathcal{F}, \mathcal{G} \subseteq 2^{\{1,\ldots,n\}}$ are Sperner hypergraphs and consist respectively of all minterms and maxterms of f (cf. [Weg87]). Note that by definition, a maxterm of f is a minimal set that contains at least one variable from every minterm of f and hence $\mathcal{G}$ consists of all minimal transversals of $\mathcal{F}$ i. e., $\mathcal{G} = \mathcal{F}^d$. It follows that the following problem is equivalent to hypergraph dualization.

Problem DNFTOCNF

Input: A monotone Boolean function f given as its irredundant DNF.

Output: The irredundant CNF of f.

The *dual* of a monotone Boolean function f is denoted by f^d and defined as the function $f^d(x) \overset{\text{def}}{=} \neg f(\neg x)$. Note that by definition, $(f^d)^d = f$ and that a maxterm (resp. minterm) of f is a minterm (resp. maxterm) of f^d. Hence the following problem is also equivalent to DUALIZATION.

Problem BOOLEANDUAL

Input: A monotone Boolean function f given as its irredundant DNF.

Output: The irredundant DNF of f^d.

We define the *readability* of a monotone Boolean function f to be the minimum integer k such that there exists an $\wedge$-$\vee$-formula equivalent to f in which each variable appears at most k times. Moreover, we define the size of $\wedge$-$\vee$-formula to be total number of occurrences of variables in it.

2.5.2 Frequent Sets in Databases

Consider the problem of finding all (inclusion-wise) maximal/minimal collections of items that are frequently/infrequently bought together by customers in a supermarket. This information for example can be used to optimize the layout of products in supermarkets and to better predict the trends in demand of certain products. More precisely, let $\mathcal{D} \in \{0,1\}^{m \times n}$ be a binary matrix whose rows represent the subsets of items purchased by different customers in a supermarket. For a given integer $t \geq 0$, a subset of items is said to be *t-frequent* if at least t rows (transactions) of $\mathcal{D}$ contain it, and otherwise is said to be *t-infrequent*. Finding frequent item-sets is an essential problem in finding the so-called *association rules* in data mining applications [AIS93].

See, e.g [AMS$^+$96, BGKM02] for other applications of minimal infrequent and maximal frequent sets in data mining.

By monotonicity, it is enough to find the *border* which is defined by the minimal t-infrequent and maximal t-frequent sets.

Problem FREQUENTSETS

Input: A database $\mathcal{D} \in \{0,1\}^{m \times n}$ and a parameter t.

Output: The family of all maximal t-frequent sets of $\mathcal{D}$.

Problem INFREQUENTSETS

Input: A database $\mathcal{D} \in \{0,1\}^{m \times n}$ and a parameter t.

Output: The family of all minimal t-infrequent sets of $\mathcal{D}$.

While it was shown in [BGKM02] that finding maximal frequent sets is an NP-hard problem, finding the minimal t-infrequent sets turns out to be polynomially equivalent with the *hypergraph transversal problem*.

Chapter 3

Fixed Parameter Algorithms

In this chapter we present fixed-parameter algorithms for the hypergraph transversal problem with the number of edges of the hypergraphs, the minimum integer p such that the input hypergraph is p-degenerate, and the maximum vertex complementary degree as our parameters. We conclude with briefly mentioning the FPT results for DUAL as well as the related problems of generating all maximal independent sets of a hypergraph and all maximal frequent sets where parameters bound the intersections or unions of edges.

3.1 Introduction

Briefly, a parameterized problem with parameter k is *fixed-parameter tractable* if it can be solved by an algorithm running in time $O(f(k) \cdot poly(n))$, where f is a function depending on k only, n is the size of the input, and $poly(n)$ is any polynomial in n. The class FPT contains all fixed-parameter tractable problems. For more general surveys on parameterized complexity and fixed-parameter tractability we refer to the monographs of Downey and Fellows, and Niedermeier [DF99, Nie06].

In this chapter, we show that DUAL$(\mathcal{G}, \mathcal{H})$ is fixed parameter-tractable with respect to the following parameters:

- the numbers of edges $m = |\mathcal{G}|$ and $m' = |\mathcal{H}|$ (cf. Section 3.2),
- the minimum integer p such that $\mathcal{G}$ is p-degenerate (cf. Section 3.3),
- the maximum degrees of vertices in $\mathcal{G}$ and $\mathcal{H}$, i. e., $d = \max_{v \in V} |\{G \in \mathcal{G} \,:\, v \in G\}|$, $d' = \max_{v \in V} |\{H \in \mathcal{H} \,:\, v \in H\}|$ (cf. Section 3.3),
- the maximum complementary degrees $q = \max_{v \in V} |\{G \in \mathcal{G} \,:\, v \notin G\}|$ and $q' = \max_{v \in V} |\{H \in \mathcal{H} \,:\, v \notin H\}|$ (cf. Section 3.4), and
- the maximum c such that $|G_1 \cup G_2 \cup \cdots \cup G_k| \geq n - c$, for any $G_1, \ldots, G_k \in \mathcal{G}$ where k is a constant—and the symmetric parameter c' with respect to $\mathcal{H}$ (cf. Section 3.5).

We shall prove the bounds with respect to the parameters m, d, p, q; the other symmetric bounds follow by exchanging the roles of $\mathcal{G}$ and $\mathcal{H}$. While for parameters c and c' we only state the results and point to the literature for further details. Our results for the parameters m and q improve the respective results from [Hag07]. Moreover, the bounds we get imply that for parameter size upto $\log N$, we get output polynomial algorithms, where $N = |\mathcal{G}| + |\mathcal{H}|$ is the input and output size of the instance.

In Section 3.5 we also consider the related problem of finding all maximal frequent sets, which turns out to be fixed parameter-tractable with respect to the maximum size of the intersection of any k rows of the database $\mathcal{D}$, for a constant k, thus generalizing the well-known Apriori algorithm, which is fixed-parameter with respect to the size of the largest transaction.

3.2 Number of Edges as Parameter

Let $(\mathcal{G}, \mathcal{H})$ be an instance of DUAL and let $m = |\mathcal{G}|$. We show that the problem is fixed-parameter tractable with parameter m and improve the running time of [Hag07].

Given a hypergraph $\mathcal{G} \subseteq 2^V$ and a subset $S \subseteq V$ of vertices, recall from Section 2.2 that S is a sub-transversal of $\mathcal{G}$ if there is a minimal transversal $T \in \mathcal{G}^d$ such that $T \supseteq S$.

In general, it is an NP-hard to decide if a given subset S is a sub-transversal even if $\mathcal{G}$ is a graph (see [BEGK04]). However, if $|S|$ is bounded by a constant, or if the hypergraph is read-once [Eit94], then such a check can be done in polynomial time (see [BGH98]). This observation was used to solve the hypergraph transversal problem in polynomial time for hypergraphs of bounded edge size or more generally of bounded conformality in [BEGK04].

The following lemma analyzes the runtime of a brute-force algorithm to decide the sub-transversal criterion of Proposition 2.8.

Lemma 3.1. *Given a hypergraph $\mathcal{G} \subseteq 2^V$ of size $|\mathcal{G}| = m$ and a subset $S \subseteq V$, of size $|S| = s$, checking whether S is a sub-transversal of $\mathcal{G}$ can be done in time $O(nm(m/s)^s)$.*

Proof. For every possible selection $\mathcal{G} = \{G_v \in \mathcal{G}_v(S) \mid v \in S\}$, we can check if $\mathcal{G}$ is non-blocked in $O(n|\mathcal{G}_{\bar{S}}|)$ time. Since the families $\mathcal{G}_v(S)$ are disjoint, we have $\sum_{v \in S} |\mathcal{G}_v(S)| \leq m$, and thus the arithmetic-geometric mean inequality gives for the total number of selections

$$\prod_{v \in S} |\mathcal{G}_v(S)| \leq \left(\frac{\sum_{v \in S} |\mathcal{G}_v(S)|}{s} \right)^s \leq \left(\frac{m}{s} \right)^s .$$

□

Our FPT algorithm uses the backtracking approach of Section 2.4.2 with the sub-transversal test of Lemma 3.1. Thus, we get the following bound on the running time.

Lemma 3.2. *Let $\mathcal{G} \subseteq 2^V$ be a hypergraph with $|\mathcal{G}| = m$ edges on $|V| = n$ vertices. Then all minimal transversals of $\mathcal{G}$ can be found with $O(n^2 m^2 e^{m/e})$ delay, where e is the base of the natural logarithm.*

Theorem 3.3. *Let $\mathcal{G}, \mathcal{H} \subseteq 2^V$ be two hypergraphs with $|\mathcal{G}| = m, |\mathcal{H}| = m'$ and $|V| = n$. Then $\mathcal{G}^d = \mathcal{H}$ can be decided in time $O(n^2 m^2 e^{(m/e)} \cdot m')$. Thus the problem is* FPT *with respect to the parameter m.*

Proof. We generate at most m' members of $\mathcal{G}^d$ (if there are more then obviously $\mathcal{G}^d \neq \mathcal{H}$). Assuming that hyperedges are represented by bit vectors (defined by indicator functions), we can check whether $\mathcal{H}$ is identical to $\mathcal{G}^d$ by lexicographically ordering the hyperedges of both and simply comparing the two sorted lists. The time to sort and compare m' hyperedges each one of size at most $\log n$ can be bounded by $O(m' \log m' \log n)$. □

As a side remark we note an interesting implication of Lemma 3.2. Let $\mathcal{G}$ be a hypergraph with $|\mathcal{G}| \leq c \ln n$ for a constant c, then Lemma 3.2 finds all its minimal transversals with polynomial delay $O(n^{2+c/e} \ln^2 n)$ improving the previous best bound of $O(n^{6+2c/\log_2 e})$ by Makino [Mak03], where e is the base of the natural logarithm.

3.3 Hypergraph Degeneracy as Parameter

Let $\mathcal{G} \subseteq 2^V$ be a p-degenerate hypergraph. We show that $\textsc{Dual}(\mathcal{G}, \mathcal{H})$ is fixed-parameter tractable with parameter p (a result which follows by similar techniques, but with weaker bounds, from [EGM03]).

As shown in Section 2.1, a p-degenerate hypergraph induced an ordering of vertices $v_1, \ldots, v_n$ such that for $1 \leq i \leq n$, the degree of a vertex v_i in the hypergraph $\mathcal{G}_{\{v_i,\ldots,v_n\}}$ is at most p. We apply the incremental technique of Section 2.4.3 to dualize $\mathcal{G}$ using the reverse ordering $v_n, \ldots, v_1$. To this end, let $\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_n$ be a partition of hypergraph $\mathcal{G}$ defined as $\mathcal{G}_i = \{G \in \mathcal{G} \; : \; G \ni v_i, G \subseteq \{v_i, \ldots, v_n\}\}$. By definition the size of each set $\mathcal{G}_i$ in this partition is bounded by p. This observation yields a simple FPT algorithm by essentially combining the technique of the previous section with the incremental method of Section 2.4.3. Recall that the incremental method generates $\mathcal{G}^d$ in time $O\left(|\mathcal{G}^d| \left(\Gamma_2 + n|\mathcal{G}||\mathcal{G}^d|\right)\right)$ where Γ_2 is the time required to solve smaller subproblems which in our case comprises of at most p edges. Using the FPT algorithm of previous section to solve these smaller instances, we get the following result.

Lemma 3.4. *Let $\mathcal{G} \subseteq 2^V$ be a p-degenerate hypergraph on $|V| = n$ vertices. Then all minimal transversals of $\mathcal{G}$ can be found in time $O\left(n^2 m'^2 \cdot \left(np^2 e^{p/e} + m\right)\right)$, where $m = |\mathcal{G}|$ and $m' = |\mathcal{G}^d|$.*

Theorem 3.5. *Let $\mathcal{G} \subseteq 2^V$ be a p-degenerate hypergraph on $|V| = n$ vertices. Then* $\textsc{Dual}(\mathcal{G}, \mathcal{H})$ *is fixed parameter tractable with respect to the parameter p.*

Since hypergraphs with maximum degree d are also d-degenerate as mentioned in Section 2.1, the above theorem implies that hypergraph dualization is also fixed parameter tractable with respect to parameter d.

3.4 Vertex complementary degree as parameter

For a hypergraph $\mathcal{G} \subseteq 2^V$ and a vertex $v \in V$, consider the number of edges in $\mathcal{G}$ not containing v for some vertex $v \in V$. Let q be maximum such number, i. e., $q = \max_{v \in V} |\{G \in \mathcal{G} \ : \ v \notin G\}|$. We show that $\textsc{Dual}(\mathcal{G}, \mathcal{H})$ is fixed-parameter tractable with parameter q and improve the running time of [Hag07].

We use Proposition 2.16 to decompose the problem into two subproblems $(\mathcal{G}_{V \setminus v}, \mathcal{H}^{V \setminus v})$ and $(\mathcal{G}^{V \setminus v}, \mathcal{H}_{V \setminus v})$ not containing a given vertex $v \in V$. Note that the hypergraph $\mathcal{G}_{V \setminus v}$ has at most q edges. This observation leads to a recursive FPT algorithm for vertex complementary degrees as parameter, by applying Proposition 2.16 at each step, solving the subproblem $(\mathcal{G}_{V \setminus v}, \mathcal{H}^{V \setminus v})$ by applying Theorem 3.3, and recursing on the subproblem $(\mathcal{G}^{V \setminus v}, \mathcal{H}_{V \setminus v})$. Since at least one vertex is reduced at each step of the algorithm, there are at most $n = |V|$ recursive steps.

Theorem 3.6. *Let $\mathcal{G}, \mathcal{H} \subseteq 2^V$ be two hypergraphs with $|\mathcal{G}| = m, |\mathcal{H}| = m'$ and $|V| = n$. Let $q = \max_{v \in V} |\{G \in \mathcal{G} \ : \ v \notin G\}|$. Then $\mathcal{G}^d = \mathcal{H}$ can be decided in time $O(n^3 q^2 e^{(q/e)} \cdot m')$.*

3.5 Results Based on the Apriori Technique

Gunopulos et al. [GKM$^+$03] showed (Theorem 23, page 156) that generating minimal transversals of hypergraphs $\mathcal{G}$ with edges of size at least $n - c$ can be done in time $O(2^c poly(n, m, m'))$, where $n = |V|$, $m = |\mathcal{G}|$ and $m' = |\mathcal{G}^d|$. This is a fixed-parameter algorithm for c as parameter. Furthermore, this result shows that the minimal transversals can be generated in polynomial time for $c \in O(\log n)$. The computation is done by an Apriori (level-wise) algorithm [AS94].

In the following, we briefly mention the results based on apriori technique without giving the proofs. See [EHR08] and the dissertation of Hagen [Hag08] for details.

Let k and c be two positive integers. We consider hypergraphs $\mathcal{G} \subseteq 2^V$ satisfying the following condition:

(C1) Any k distinct maximal independent sets $I_1, \ldots, I_k$ of $\mathcal{G}$ intersect in at most c vertices, i. e., $|I_1 \cap \cdots \cap I_k| \leq c$.

We note that the above property can be verified in polynomial time (see, [EHR08]).

Using the apriori approach, we obtain that we can compute all the maximal independent sets in time $O(\min\{2^c(m')^k poly(n,m), e^{k/e}n^{c+1}poly(m,m')\})$ if any k distinct maximal independent sets of a hypergraph $\mathcal{G}$ intersect in at most c vertices.

Theorem 3.7. *If any k distinct maximal independent sets of a hypergraph $\mathcal{G}$ intersect in at most c vertices, then in $O(\min\{2^c(m')^k poly(n,m), e^{k/e}n^{c+1}poly(m,m')\})$ time, all maximal independent sets can be computed, where $n = |V|$, $m = |\mathcal{G}|$ and $m' = |\mathcal{G}^{dc}|$.*

Equivalently, if the union of any k distinct minimal transversals has size at least $n - c$, then all minimal transversals can be computed in the same time bound.

Corollary 3.8. *Let $\mathcal{G} \subseteq 2^V$ be a hypergraph on $n = |V|$ vertices, and k, c be positive integers.*

(i) If any k distinct minimal transversals of $\mathcal{G}$ have a union of at least $n - c$ vertices, we can compute all minimal transversals in $O(\min\{2^c(m')^k poly(n,m),\ e^{k/e}n^{c+1}poly(m,m')\})$ time, where $m = \mathcal{G}$ and $m' = |\mathcal{G}^d|$.

(ii) If any k distinct hyperedges of $\mathcal{G}$ have a union of at least $n-c$ vertices, we can compute all minimal transversals in time $O(\min\{2^c m^k poly(n,m'),\ e^{k/e}n^{c+1}poly(m,m')\})$, where $m = \mathcal{G}$ and $m' = |\mathcal{G}^d|$.

And again using the same idea, we obtain that the maximal frequent sets of an $m \times n$ database can be computed in $O(2^c(nm')^{2^{k-1}+1}poly(n,m))$ time if any k rows of it intersect in at most c items, where m' is the number of such sets.

Theorem 3.9. *If any k distinct maximal frequent sets intersect in at most c items, we can compute all maximal frequent sets in $O(2^c(nm')^k poly(n,m))$ time, where m' is the number of maximal frequent sets.*

Corollary 3.10. *If any k distinct transactions intersect in at most c items, then all maximal frequent sets can be computed in time $O(2^c(nm')^{2^{k-1}+1}poly(n,m))$, where m' is the number of maximal frequent sets.*

Note that for $c \in O(\log n)$ and $k \in O(1)$ we have incremental polynomial-time algorithms for all four problems.

Chapter 4

r-Exact Hypergraphs

Let $\mathcal{H} \subseteq 2^V$ be a hypergraph on vertex set V. Recall that $\mathcal{H}$ is called r-exact, if any minimal transversal of $\mathcal{H}$ intersects any hyperedge of $\mathcal{H}$ in at most r vertices. This class includes several interesting examples from geometry, e.g., circular-arc hypergraphs ($r = 2$), hypergraphs defined by sets of axis-parallel lines stabbing a given set of α-fat objects ($r = O(\alpha)$), and hypergraphs defined by sets of points contained in translates of a given cone in the plane ($r = 2$). For constant r, we give a polynomial-time algorithm for the duality testing problem of a pair of r-exact hypergraphs. This result implies that minimal hitting sets for the above geometric hypergraphs can be generated in incremental polynomial time.

4.1 Introduction

Given an integer $r \geq 1$, recall from Chapter 2 that a hypergraph $\mathcal{H} \subseteq 2^V$ is called *r-exact* if

$$|H \cap T| \leq r, \text{ for all } H \in \mathcal{H} \text{ and } T \in \mathcal{H}^d. \tag{4.1}$$

As we shall see later, these hypergraphs can be recognized in polynomial time. Clearly, if $\mathcal{H}$ satisfies (4.1) then so do the hypergraphs $\mathcal{H}_S$ and $\mathcal{H}^S$, for any $S \subseteq V$. Note that this class of hypergraphs includes the case when $\max\{|H| : H \in \mathcal{H}\} \leq r$ or $\max\{|T| :$

$T \in \mathcal{H}^d\} \leq r$.

The class of hypergraphs satisfying (4.1) with $r = 1$, are called *exact* or *read-once* hypergraphs, and are related to a class of monotone Boolean functions also called read-once functions. We will look at read-once functions in detail in Section 6.2.

For a read-once hypergraph $\mathcal{H}$, the problem of finding $\mathcal{H}^d$ is known to be solvable with polynomial delay, using a simple backtracking approach [Eit94]. However, this technique does not seem to generalize for $r \geq 2$. Using a more sophisticated technique, we show in Section 4.3 that the problem can still be solved in incremental polynomial-time, for any hypergraph satisfying (4.1). The best previously known result for this class was quasi-polynomial $\mathrm{poly}(n, |\mathcal{H}^d|)|\mathcal{H}|^{O(\log |\mathcal{H}|)}$ [KBEG07b] (which gives in fact a global parallel algorithm running in polylogarithmic time and requiring a quasi-polynomial number of processors). In this chapter, we prove the following theorem.

Theorem 4.1. *Let $\mathcal{H}$ be an r-exact hypergraph with m edges and n vertices, and k be a given positive integer. Then for $r = O(1)$, we can find k minimal transversals of $\mathcal{H}$ in time* $\mathrm{poly}(n, m, k)$.

As consequences of Theorem 4.1, we obtain incremental polynomial-time algorithms for finding:

- all minimal hitting sets, and all minimal set covers, for a circular-arc hypergraph (see, e.g., [FS91]). This generalizes known results for interval hypergraphs [EGM02, BEGK04];
- all minimal hitting sets, and all minimal set covers, for a hypergraph defined by a set of points and given translates of a certain cone in the plane;
- all minimal subsets of a given set of axis-parallel hyperplanes, hitting a set of comparable fat objects in $\mathbb{R}^d$, for fixed d (see, e.g., [GIK02, KS06] for the corresponding optimization problems).

We now note that testing a hypergraph $\mathcal{H}$ for (4.1) can be done polynomial time, if r is a constant.

Proposition 4.2. *Given a hypergraph $\mathcal{H} \subseteq 2^V$ and a constant r, whether $\mathcal{H}$ is an r-exact hypergraph can be checked in time $O(n^{r+2}m^{r+3})$, where $n = |V|$ and $m = |\mathcal{H}|$.*

Proof. $\mathcal{H}$ satisfies (4.1) if and only if for every edge $H \in \mathcal{H}$ and every subset $X \subseteq H$ of size $|X| = r + 1$, X is not a sub-transversal to $\mathcal{H}$. For a single subset X of size $r + 1$, the latter condition can be checked in $O(nm^{r+2})$ time by directly checking the condition of Proposition 2.8 for a total of $O(\sum_{H\in\mathcal{H}} \binom{|H|}{r+1} nm^{r+2})$ time. □

The rest of the chapter is organized as follows. In Section 4.2, we give the details of the geometric applications listed above. Finally, we prove Theorem 4.1 in Section 4.3.

4.2 Applications in geometry

In this section we give some examples of hypergraphs satisfying (4.1) from geometry.

4.2.1 Circular-arc hypergraphs

Let C be a circle in the plane, and $V = \{p_0, \ldots, p_{n-1}\}$ be a given set of points on C. Assume that the points are ordered in clockwise order around C. Let $\mathcal{H}$ be a hypergraph consisting of hyperedges that are defined by consecutive elements of V (that is, arcs or intervals on the circle, see Figure 4.1). Note that if $\mathcal{H}$ is defined by sets of intervals on *the line*, then $\mathcal{H}$ is both 1-degenerate and 2-Helly (its transpose is 2-conformal), and hence both problems of finding all minimal hitting sets and of finding all minimal set covers can be solved in polynomial time [BEGK04, EGM02]. However, if $\mathcal{H}$ is defined by arcs on a circle, then it is not generally k-degenerate, as shown by the following example: Let $\mathcal{H}(k, n)$ be the hypergraph of uniform intervals of length k on a ring of $n = k^2$ vertices:

$$\mathcal{H}(k, n) = \{[i : i + k - 1](\text{mod } n) | i = 0, \ldots, n - 1\}.$$

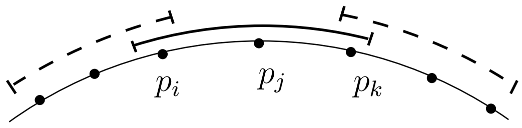

Figure 4.1: In a circular-arc hypergraph, every minimal transversal intersects every interval at most twice.

Then each vertex has degree k. Furthermore, it is not k-Helly either for any k, as shown by following hypergraph consisting of $k+1$ intervals on a ring of $k+1$ vertices:

$$\mathcal{H}(k) = \{[i : i+k-1](\text{mod } k+1) | i = 0, \ldots, k\}.$$

It is easy to see that any k intervals in $\mathcal{H}(k)$ intersect at a common point whereas the intersection of all intervals is empty.

Nevertheless, we show in the following that if $\mathcal{H}$ is Sperner circular-arc hypergraph, then every minimal transversal hits every edge in $\mathcal{H}$ at most twice.

Proposition 4.3. *Let $\mathcal{H}$ be a Sperner circular-arc hypergraph. Then $\mathcal{H}$ is 2-exact.*

Proof. Let T be a minimal transversal of $\mathcal{H}$, and suppose that $|T \cap H| \geq 3$ for some $H \in \mathcal{H}$. Consider any three points $p_i, p_j, p_k \in T \cap H$, such that $i < j < k$ (mod n). Since T is a minimal transversal, there exists $H' \in \mathcal{H}$ such that $H' \cap T = \{p_j\}$. But then H' contains neither p_i nor p_k, and hence $H' \subset H$, in contradiction to the fact that $\mathcal{H}$ is Sperner (see Figure 4.1). $\square$

Since the transpose of a circular-arc hypergraph $\mathcal{H}$ is also circular-arc, we obtain the following result from Theorem 4.1 and Proposition 4.3.

Corollary 4.4. *Let V be a set of points on a circle C and $\mathcal{A}$ be a set of circular arcs on C. Then both problems of finding all minimal sets of points that hit all the arcs in $\mathcal{A}$, and of finding all minimal sets of arcs that cover all the points can be solved in incremental polynomial time.*

Geometrically, the same class can be realized by taking a set of points in convex position in the plane and defining each edge as a subset of points which lie in some half-space. However, this does not generalize to higher dimensions.

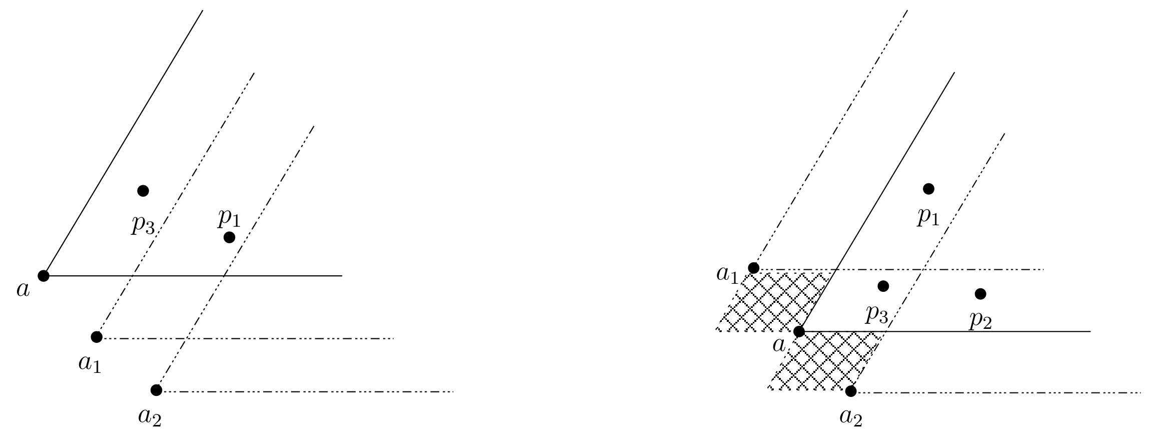

Figure 4.2: The two intersection configurations of the cones $C(a_1)$, $C(a_2)$ and $C(a)$.

4.2.2 Translates of cones in $\mathbb{R}^2$

Let $C = \{x \in \mathbb{R}^2 : \ a_1^T x \leq 0, \ a_2^T x \leq 0\}$ be a cone in the plane, where $a_1, a_2 \in \mathbb{R}^2$. For a point $a \in \mathbb{R}^2$, define $C(a)$ to be the translate of C with apex a, i.e., $C(a) = \{x + a \ : \ x \in C\}$. Given a set of apexes $A \subseteq \mathbb{R}^2$ and a set of points $V \subseteq \mathbb{R}^2$, we define the hypergraph $\mathcal{H}(C, A, V) = \{V \cap C(a) : a \in A\}$, each hyperedge of which is defined by the subset of V that lies inside some translate of C (see Figure 4.2).

Proposition 4.5. *Let C be a cone in the plane and $A, V \subseteq \mathbb{R}^2$. If $\mathcal{H} = \mathcal{H}(C, A, V)$ is Sperner, then $\mathcal{H}$ is 2-exact.*

Proof. Note that for $a, a' \in A$, $C(a) \subseteq C(a')$ if and only if $a \in C(a')$. Let T be a minimal a transversal of $\mathcal{H}$, and suppose that $|T \cap C(a)| \geq 3$ for some $a \in A$. Consider any three points $p_1, p_2, p_3 \in T \cap C(a)$. By minimality of T, there are apexes $a_1, a_2, a_3 \in A$ such that $C(a_i) \cap \{p_1, p_2, p_3\} = \{p_i\}$, for $i = 1, 2, 3$. We may assume without loss of generality that $C(a_1)$ and $C(a_2)$ intersect $C(a)$ as shown in Figure 4.2, right, since otherwise (as shown in Figure 4.2, left), we have $C(a_2) \cap C(a) \subset C(a_1) \cap C(a)$, or vice versa. We observe that the apex a_3 must lie in one of the two shaded regions, for otherwise $C(a_3) \subseteq C(a)$, $C(a_3) \supseteq C(a_1)$, or $C(a_3) \supseteq C(a_2)$, in contradiction to the fact that $\mathcal{H}$ is Sperner. But if a_3 is contained in the first shaded region then $p_1 \in C(a_3)$, and if it is contained in the second, then $p_2 \in C(a_3)$, and in both cases we get a contradiction. $\square$

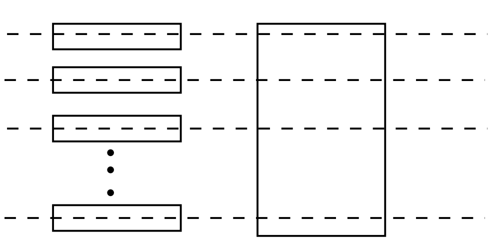
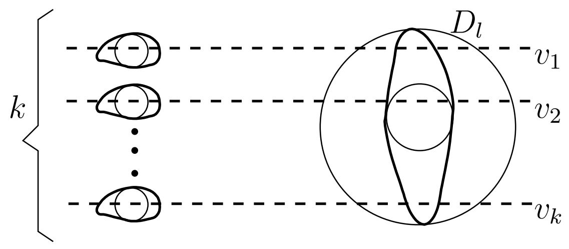

Figure 4.3: Left: If we do not require objects to be fat then intersection can be unbounded. Right: Illustration for Proposition 4.7.

Again, the transpose of $\mathcal{H}(C, A, V)$ can also be realized by translates of a cone, as implied by a result of Laue [Lau08]. Thus we get the following corollary from Theorem 4.1 and Proposition 4.5.

Corollary 4.6. *Let V be a set of points in the plane and $\mathcal{C}$ be a set of cones that are translates of a fixed cone in the plane. Then both problems of finding all minimal sets of points hitting every cone, and of finding all minimal sets of cones that covers all points can be solved in incremental polynomial time.*

It is not difficult to see that Proposition 4.5 does not generalize to higher dimensions.

4.2.3 Stabbing fat objects in $\mathbb{R}^d$

Let $\mathcal{O} \subseteq \mathbb{R}^d$ be a d-dimensional object. For $\alpha \geq 1$, $\mathcal{O}$ is said to be *α-fat* [Kal97] if the ratio between the diameter of the smallest ball containing $\mathcal{O}$ and the largest ball contained in $\mathcal{O}$ is at most α. For instance, for a ball $\alpha = 1$, for a hypercube $\alpha = \sqrt{d}$, and for a line segment $\alpha = \infty$. The diameter of $\mathcal{O}$, denoted $\mathrm{diam}(\mathcal{O})$, is defined as the diameter of the smallest enclosing ball.

Let $\mathcal{C}$ be a given collection of connected α-fat objects with ρ-comparable diameters, i.e., $\mathrm{diam}(\mathcal{O}) \leq \rho \cdot \mathrm{diam}(\mathcal{O}')$, for all $\mathcal{O}, \mathcal{O}' \in \mathcal{C}$ and some constant $\rho \geq 1$. Given a set of axis parallel hyperplanes V, we are interested in finding all minimal subsets of hyperplanes from V that stab every object in $\mathcal{C}$. Define the hypergraph $\mathcal{H}(\mathcal{C}, V) = \{\{v \in V : v \text{ stabs } \mathcal{O}\} : \mathcal{O} \in \mathcal{C}\}$. We show that $\mathcal{H}(\mathcal{C}, V)$ is $O(1)$-exact, for fixed d, ρ, and α.

Proposition 4.7. *Let $\mathcal{C}$ be a collection of connected α-fat objects in $\mathbb{R}^d$ with ρ-comparable diameters, and V a set of axis parallel hyperplanes. Then $\mathcal{H}(\mathcal{C}, V)$ is $2\alpha\rho d$-exact.*

Proof. Fix an axis $\vec{x}$, let T be a minimal transversal, and denote by $T_{\vec{x}}$ the hyperplanes in T which are perpendicular to $\vec{x}$. Assume that some object $\mathcal{O} \in \mathcal{C}$ is hit by k members of $T_{\vec{x}}$, say, $T_{\vec{x}} \cap \{v \in V : v \text{ stabs } \mathcal{O}\} = \{v_1, \ldots, v_k\}$ (see Figure 4.3, right). Let $D_l = \mathrm{diam}(\mathcal{O})$ and let D_0 be the minimum diameter of enclosed spheres among our collection of objects $\mathcal{C}$. By minimality of T, there exist objects $\mathcal{O}_i \in \mathcal{C}$ such that $T \cap \{v \in V : v \text{ stabs } \mathcal{O}_i\} = \{v_i\}$ for $i = 1, \ldots, k$. Assume w.l.o.g. that $v_1, v_2, \ldots, v_k$ are ordered along $\vec{x}$, as in Figure 4.3, right. Observe that $\mathcal{O}_i, \mathcal{O}_{i+2}$ do not intersect for $i = 1, \ldots, k-2$, since otherwise v_{i+1} stabs either $\mathcal{O}_i$, or $\mathcal{O}_{i+2}$, in contradiction to our assumption. Therefore, we can conclude that $D_l \geq \frac{k}{2} D_0$. Also by the assumption that objects in $\mathcal{C}$ are α-fat and ρ-comparable we have $D_l \leq \rho\alpha D_0$. Comparing the two inequalities, we get $k \leq 2\rho\alpha$. By applying this argument to every principal axis, we get the result. $\square$

Note that fatness is a necessary condition for Proposition 4.7 to hold (see Figure 4.3, left).

Corollary 4.8. *Let $\mathcal{C}$ be a collection of α-fat objects in $\mathbb{R}^d$ with ρ-comparable diameters, and V a set of axis parallel hyperplanes. Then for constant α, ρ, d, all minimal sets of hyperplanes from V stabbing every object in $\mathcal{C}$ can be found in incremental polynomial time.*

4.3 Dualization of r-exact hypergraphs

To prove Theorem 4.1, we use similar decompositions as the ones used in [KBEG07b]. However, to get a polynomial time algorithm, one uses the fact that duality testing of two hypergraphs is a symmetric operation with respect to the two hypergraphs. Combining this with a simple decomposition rule from Proposition 2.16, we can derive our result.

4.3.1 Decompositions

Let V be a finite set, $r \geq 2$ be a positive constant, and $\mathcal{H} \subseteq 2^V$ be a hypergraph satisfying (4.1). As it follows from Proposition 2.9, it is enough to solve $\text{DUAL}(\mathcal{H}, \mathcal{G})$ to prove Theorem 4.1.

Recall from Section 2.4.5 that a family of sets $\{S_1, \dots, S_k\} \subseteq 2^V$ is a full cover for $\mathcal{H}$ if for every $H \in \mathcal{H}$ there is an $i \in [k]$ such that $H \subseteq S_i$. The main ingredient of our algorithm is the following decomposition, used (implicitly) in [KBEG07b].

Lemma 4.9. *Let $\mathcal{H}, \mathcal{G} \subseteq 2^V$ be hypergraphs satisfying Equation* (2.1)*, and $\{S_1, \dots, S_k\}$ be a full cover of $\mathcal{H}$. Then $(\mathcal{H}, \mathcal{G})$ are dual if and only if*

(i) for all $X \in \bigwedge_{i \in [k]} \mathcal{G}^{S_i}$ there is a $G \in \mathcal{G}$, such that $X \supseteq G$, and

(ii) $(\mathcal{H}_{S_i}, \mathcal{G}^{S_i})$ are dual for all $i \in [k]$.

Proof. Suppose that (i) does not hold. Then there is an $X \in \wedge_{i \in [k]} \mathcal{G}^{S_i}$ such that $X \not\supseteq G$ for all $G \in \mathcal{G}$. We note that X is a transversal to $\mathcal{H}$ (since the family $\{S_1, \dots, S_k\}$ is a full cover and $(\mathcal{H}, \mathcal{G})$ satisfy (2.1)). This implies that X satisfies (2.4) with respect to $(\mathcal{H}, \mathcal{G})$ and thus witnessing the fact that $\mathcal{H}$ and $\mathcal{G}$ are not dual to each other.

Suppose that (ii) does not hold, i.e., for some $i \in [k]$, there is an $X \subseteq S_i$ such that X satisfies (2.4) with respect to the pair $(\mathcal{H}_{S_i}, \mathcal{G}^{S_i})$. Then $X \cup (V \setminus S_i)$ satisfies (2.4) with respect to $(\mathcal{H}, \mathcal{G})$.

Suppose now that there is an X that satisfies (2.4) with respect to $(\mathcal{H}, \mathcal{G})$. If (ii) holds, then for every $i \in [k]$, there is a $G_i \in \mathcal{G}^{S_i}$, such that $G_i \subseteq X \cap S_i$. But then $X \supseteq \cup_{i \in [k]} G_i$, and hence there is a $Y \in \wedge_{i \in [k]} \mathcal{G}^{S_i}$ such that $Y \subseteq X$. But (i) implies that $Y \supseteq G$ and hence $X \supseteq G$ for some $G \in \mathcal{G}$, a contradiction. □

A symmetric version of Lemma 4.9 can be obtained by exchanging the roles of $\mathcal{H}$ and $\mathcal{G}$ and is stated as follows.

Lemma 4.10. *Let $\mathcal{H}, \mathcal{G} \subseteq 2^V$ be hypergraphs satisfying Equation* (2.1)*, and $\{S_1, \dots, S_k\}$ be a full cover of $\mathcal{G}$. Then $(\mathcal{H}, \mathcal{G})$ are dual if and only if* *(i) for all $X \in \bigwedge_{i\in[k]} \mathcal{H}^{S_i}$ there is a $H \in \mathcal{H}$, such that $X \supseteq H$, and*

(ii) $(\mathcal{H}^{S_i}, \mathcal{G}_{S_i})$ are dual for all $i \in [k]$.

Our algorithm also makes use of Proposition 2.16 to partition the problem into two subproblems.

The above lemmas allow us to reduce a given duality testing problem into smaller sub-problems. We use (4.1) to define full covers of $\mathcal{H}$ and $\mathcal{G}$. For an edge G of $\mathcal{G}$, consider its partition into $r+1$ subsets $G_1 \dots G_{r+1}$. Since any edge H of $\mathcal{H}$ intersects G in at most r vertices, there would be at least one G_i such that $H \cap G_i$ is empty. In other words, if we define $S_i = V \setminus G_i$ then for every edge H of $\mathcal{H}$ there is an $i \in [r+1]$ such that $H \subseteq S_i$. However, the covering we get in this way does not necessarily yield smaller subproblems and may not be balanced and so not much useful for our divide-and-conquer approach. In the following we present a method to get around these issues.

For a subset of vertices S, let $\mathcal{H}(S)$ denote the set of hyperedges that have at least one vertex in S, i.e., $\mathcal{H}(S) = \{H \in \mathcal{H} \mid H \cap S \neq \emptyset\}$. Let $0 < \epsilon_1 < \epsilon_2 < 1/r$ be positive constants. Assume that there is an edge G of $\mathcal{G}$ such that all of its vertices have low degrees in $\mathcal{H}$, i.e., $\deg_{\mathcal{H}}(v) < \epsilon_1|\mathcal{H}|,\ \forall v \in G$. For an arbitrary ordering $v_1, \dots, v_t$ of the vertices of G, let us find the indices $i_0 = 0 < i_1 < \dots < i_{r+1} = t$, such that $|\mathcal{H}(\{v_{i_{j-1}+1}, \dots, v_{i_j}\})| \le \epsilon_2|\mathcal{H}|$ and $|\mathcal{H}(\{v_{i_{j-1}+1}, \dots, v_{i_j+1}\})| > \epsilon_2|\mathcal{H}|$ hold for $j = 1, \dots, r$. Note that such indices exist since G is a transversal to $\mathcal{H}$, each vertex in G has degree less than $\epsilon_1|\mathcal{H}|$, and $\epsilon_1 < \epsilon_2 < 1/r$. Let us define a partition of G by breaking at these indices, i.e., $G_j = \{v_{i_{j-1}+1}, \dots, v_{i_j}\}$ and let $S_j = V \setminus G_j$, for $j = 1, \dots, r+1$.

We will show that $\{S_1, \dots, S_{r+1}\}$ is a full cover of $\mathcal{H}$ with some nice properties. Notice that for every $H \in \mathcal{H}$ we have $H \subseteq S_j$ for some $j \in [r+1]$, since $|H \cap G| \le r$ and so there must exist G_j such that $H \cap G_j = \emptyset$. Also note that $|\mathcal{H}_{S_j}| \le \epsilon|\mathcal{H}|$, for $j = 1, \dots, r+1$, where $\epsilon = \max\{r\epsilon_2, 1-(\epsilon_2-\epsilon_1)\} \in (0,1)$. Indeed, for $j = 1, \dots, r$, we have $|\mathcal{H}_{S_j}| < (1-(\epsilon_2-\epsilon_1))|\mathcal{H}|$, since $\epsilon_2|\mathcal{H}| < |\mathcal{H}(G_j \cup \{v_{i_j+1}\})| \le |\mathcal{H}(G_j)| + \deg_{\mathcal{H}}(v_{i_j+1}) < |\mathcal{H}(G_j)| + \epsilon_1|\mathcal{H}|$. Moreover, $|\mathcal{H}_{S_{r+1}}| \le |\mathcal{H}(G_1 \cup \ldots \cup G_r)| \le \sum_{j=1}^{r} |\mathcal{H}(G_j)| \le r\epsilon_2|\mathcal{H}|$. In summary, we have proved the following Lemma.

Lemma 4.11. *Let $\mathcal{H}, \mathcal{G}$ be r-exact hypergraphs satisfying* (2.1) *and let $0 < \epsilon_1 < \epsilon_2 < 1/r$ be positive constants. If there is an edge G in $\mathcal{G}$ such that $deg_{\mathcal{H}}(v) < \epsilon_1|\mathcal{H}|$ for all $v \in G$, then we can find a full cover $\{S_1, \ldots, S_{r+1}\}$ of $\mathcal{H}$. Furthermore $|\mathcal{H}_{S_j}| \leq \epsilon|\mathcal{H}|$ for all $j = 1, \ldots, r+1$, where $\epsilon = \max\{r\epsilon_2, 1-(\epsilon_2-\epsilon_1)\}$.*

By symmetry we get a similar result by reversing the roles of $\mathcal{H}$ and $\mathcal{G}$.

Lemma 4.12. *Let $\mathcal{H}, \mathcal{G}$ be r-exact hypergraphs satisfying* (2.1) *and let $0 < \epsilon_1 < \epsilon_2 < 1/r$ be positive constants. If there is an edge H in $\mathcal{H}$ such that $deg_{\mathcal{G}}(v) < \epsilon_1|\mathcal{G}|$ for all $v \in H$, then we can find a full cover $\{S_1, \ldots, S_{r+1}\}$ of $\mathcal{G}$. Furthermore $|\mathcal{G}_{S_j}| \leq \epsilon|\mathcal{G}|$ for all $j = 1, \ldots, r+1$, where $\epsilon = \max\{r\epsilon_2, 1-(\epsilon_2-\epsilon_1)\}$.*

The only case that remains is when both the preconditions of Lemma 4.11 and Lemma 4.12 do not apply. The following lemma will cover this case.

Lemma 4.13. *Let $\mathcal{H}, \mathcal{G}$ be hypergraphs satisfying* (2.1) *and let $0 < \epsilon_1 < 1$ be a positive constant. Let $L = \{v \in V : \deg_{\mathcal{H}}(v) \geq \epsilon_1|\mathcal{H}|\}$ and $L' = \{v \in V : \deg_{\mathcal{G}}(v) \geq \epsilon_1|\mathcal{G}|\}$. If $G \cap L \neq \emptyset$ for all $G \in \mathcal{G}$ and $H \cap L' \neq \emptyset$ for all $H \in \mathcal{H}$ then either $\mathcal{H}$ and $\mathcal{G}$ are not dual or there exists a vertex $v \in V$ with high degrees in both $\mathcal{H}$ and $\mathcal{G}$, i.e., $\deg_{\mathcal{H}}(v) \geq \epsilon_1|\mathcal{H}|$ and $\deg_{\mathcal{G}}(v) \geq \epsilon_1|\mathcal{G}|$.*

Proof. If $L \not\supseteq H$ for all $H \in \mathcal{H}$, then $V \setminus L$ satisfies (2.4) with respect to $(\mathcal{H}, \mathcal{G})$. Similarly, if $L' \not\supseteq G$ for all $G \in \mathcal{G}$, then L' satisfies (2.4) with respect to $(\mathcal{H}, \mathcal{G})$. We assume therefore that there exist $H \in \mathcal{H}$ and $G \in \mathcal{G}$ such that $H \subseteq L$ and $G \subseteq L'$. Then by (2.1), there exists a $v \in V$ such that $\deg_{\mathcal{H}}(v) \geq \epsilon_1|\mathcal{H}|$ and $\deg_{\mathcal{G}}(v) \geq \epsilon_1|\mathcal{G}|$. □

4.3.2 Algorithm

Given a pair of r-exact hypergraphs $\mathcal{H}, \mathcal{G} \subseteq 2^V$ and two constants $\epsilon_1 < \epsilon_2 < 1/r$, an algorithm to solve DUAL$(\mathcal{H}, \mathcal{G})$ is given as Algorithm 6. If one of the sizes $|\mathcal{H}|$ or $|\mathcal{G}|$ is at most 1, then duality can be checked in polynomial time using a simple procedure **Dual-Simple**() from Section 2.4.1.1. Otherwise, the algorithm first tries to apply the decompositions of Lemma 4.11 and Lemma 4.12. If both cases do not apply, then Lemma 4.13 guarantees the existence of a high degree vertex in both hypergraphs, which we use to partition our problem by applying Proposition 2.16.

The following lemma gives a bound on the number of iterations $t(v)$ in terms of $v = |\mathcal{H}||\mathcal{G}|$, $n = |V|$ and $\epsilon = \max\{1-\epsilon_1, r\epsilon_2, 1-(\epsilon_2-\epsilon_1)\} \in (0,1)$, where $\epsilon_1 = \frac{1}{2r+1}$ and $\epsilon_2 = \frac{2}{2r+1}$ give us the best ϵ obeying the given constraints.

Lemma 4.14. $t(v) = \Theta\left(nv^{r+1}\right).$

Proof. Consider a particular iteration of the procedure. If the conditions in Step 4 or 13 becomes true, we get the recurrence

$$t(v) \leq nv^{r+1} + (r+1)t(\epsilon v), \tag{4.2}$$

while in Step 15, we get the recurrence

$$t(v) \leq nv + 2t(\epsilon v). \tag{4.3}$$

Clearly, the recurrence (4.2) dominates the recurrence (4.3) when $r \geq 1$. Applying Akra-Bazi theorem [AB98] on the recurrence (4.2) yields the stated claim. □

4.4 Conclusions

In this chapter, we have shown how to dualize r-exact hypergraphs in output polynomial time when r is a constant. We also have given some examples of geometric hypergraphs which are r-exact for a constant r. In the next chapter, we will show how to dualize a broader class of geometric hypergraphs in output polynomial time.

Algorithm 6 Dualizing r-exact hypergraphs

```
Procedure Duality($\mathcal{H}, \mathcal{G}, V, \epsilon_1, \epsilon_2$):
Input: $r$-exact hypergraphs $\mathcal{H}, \mathcal{G} \subseteq 2^V$, and positive constants $0 < \epsilon_1 < \epsilon_2 < 1/r$.
Output: Returns true if $(\mathcal{H}, \mathcal{G})$ are dual, returns a witness of non-duality otherwise.
 1: if $\min\{|\mathcal{H}|, |\mathcal{G}|\} \leq 1$ then
 2:    return Dual-Simple($\mathcal{H}, \mathcal{G}$).
 3: Let $L = \{v \in V : \deg_{\mathcal{H}}(v) \geq \epsilon_1|\mathcal{H}|\}$ and $L' = \{v \in V : \deg_{\mathcal{G}}(v) \geq \epsilon_1|\mathcal{G}|\}$.
 4: if $(\exists G \in \mathcal{G}, G \subseteq V \setminus L)$ then
 5:    Let $\{S_1, \ldots, S_{r+1}\}$ be a full cover of $\mathcal{H}$ (Lemma 4.11)
 6:    Call Duality($\mathcal{H}_{S_i}, \mathcal{G}^{S_i}, \epsilon_1, \epsilon_2$) for all $i \in [r+1]$
 7:    if any call in Step 6 fails then
 8:       return the corresponding witness of non-duality
 9:    for all $X \in \bigwedge_{i \in [r+1]} \mathcal{G}^{S_i}$ do
10:       if there is no $G \in \mathcal{G}$ such that $X \supseteq G$ then
11:          return $X$ as a witness of non-duality
12:    return true
13: else if $(\exists H \in \mathcal{H}, H \subseteq V \setminus L')$ then
14:    Repeat Steps 5-12 with the roles of $\mathcal{H}$ and $\mathcal{G}$ reversed
15: else if $(\exists v \in V$ such that $\deg_{\mathcal{H}}(v) \geq \epsilon_1|\mathcal{H}|$ and $\deg_{\mathcal{G}}(v) \geq \epsilon_1|\mathcal{G}|)$ then
16:    Call Duality($\mathcal{H}_{V \setminus v}, \mathcal{G}^{V \setminus v}, \epsilon_1, \epsilon_2$); Call Duality($\mathcal{H}^{V \setminus v}, \mathcal{G}_{V \setminus v}, \epsilon_1, \epsilon_2$)
17:    if any call in Step 16 fails then
18:       return a witness of non-duality
19:    return true
20: else
21:    return a witness of non-duality (Lemma 4.13)
```

Chapter 5

Geometric Hypergraphs

In this chapter, we present two approaches to dualize geometric hypergraphs, i. e., hypergraphs arising from geometric objects in $\mathbb{R}^d$ for a fixed d. The two general frameworks which are presented in Section 5.2 and Section 5.3 yield incremental polynomial algorithms for several types of geometric hypergraphs. While the first approach uses elementary techniques, the second one relies on simplicial partitions and cuttings, and yields global parallel dualization algorithms for a larger class of hypergraphs. In Section 5.4, we present a polynomial delay algorithm for hypergraphs defined by sets of points and half-planes in $\mathbb{R}^2$.

5.1 Introduction

Let $\mathcal{P}, \mathcal{Q}$ be two families of geometric objects in $\mathbb{R}^d$ for a fixed d. Associating a vertex with each member of $\mathcal{P}$, we define a hypergraph $(\mathcal{P}, \mathcal{Q})$ induced by these two families as $(\mathcal{P}, \mathcal{Q}) = \{\{P \in \mathcal{P} | P \cap Q \neq \emptyset\} \,|\, Q \in \mathcal{Q}\}$. In this chapter, we will often refer to hypergraphs as range spaces in accordance with the terminology in the Computational Geometry literature. Accordingly, we will refer to the vertex set as the *ground set* and the hyperedges as *ranges*. There are two natural range spaces defined by $\mathcal{P}$ and $\mathcal{Q}$ depending on whether we let $\mathcal{P}$ or $\mathcal{Q}$ form the ground set. We denote by $(\mathcal{P}, \mathcal{Q})$ the range space in which $\mathcal{P}$ is the ground set and each $Q \in \mathcal{Q}$ defines the range $\{P \in$

$\mathcal{P}|P \cap Q \neq \emptyset\}$ as defined above. Similarly, we denote by $(\mathcal{Q}, \mathcal{P})$ the range space in which the ground set is $\mathcal{Q}$ and each $P \in \mathcal{P}$ defines the range $\{Q \in \mathcal{Q}|Q \cap P \neq \emptyset\}$. Note that $(\mathcal{Q}, \mathcal{P}) = (\mathcal{P}, \mathcal{Q})^t$.

When $\mathcal{P}$ is a set of points and $\mathcal{Q}$ be a set of geometric objects then $(\mathcal{P}, \mathcal{Q})^d$ is the set of all minimal hitting sets of $\mathcal{Q}$, while $(\mathcal{Q}, \mathcal{P})^d$ is the set of all minimal set covers of $\mathcal{P}$.

In this chapter, we introduce general frameworks to solve the hypergraph transversal problem for geometric hypergraphs as introduced above. The first approach, presented in Section 5.2, uses only elementary techniques, and gives polynomial-time algorithms for the problems of hitting axis-parallel hyper-rectangles by points and stabbing connected objects by axis-parallel hyperplanes, both in $\mathbb{R}^d$ for a fixed d. In Section 5.3, we present another technique based on simplicial partitions and cuttings, to efficiently enumerate all minimal hitting-sets as well as minimal covers of hypergraphs defined by the intersection of sets of points with half-spaces (and hence balls) or more generally, polytopes with a constant number of facets all in fixed dimension $\mathbb{R}^d$. Finally, in Section 5.4 we give a polynomial delay algorithm for the special case of hypergraphs induced by half-planes and points in $\mathbb{R}^2$.

The enumeration of minimal geometric hitting sets, as the ones described above, arises in various areas such as computational geometry, machine learning, and data mining [EMG08]. Moreover, our efficient enumeration algorithms might be useful in developing exact algorithms, fixed-parameter tractable algorithms, and polynomial-time approximation schemes for the corresponding optimization problems (see, e.g., [HNW08]).

5.2 First Approach: Using Elementary Techniques

This section presents a general framework for the problem of finding all minimal hitting sets of a family of objects in $\mathbb{R}^d$ by another. It can be regarded as a generalization of the algorithms given in [EGM03, LLK80], i. e., the incremental approach of Section

2.4.3. We apply it to the following problems:

- **Hitting hyper-rectangles by points:** Given a finite set of points $\mathcal{P} \subseteq \mathbb{R}^d$ and a finite collection $\mathcal{F}$ of axis-parallel hyper-rectangles (also called orthotopes or boxes) in $\mathbb{R}^d$, find all minimal sets of points from $\mathcal{P}$ that hit every hyper-rectangle in $\mathcal{Q}$;

- **Hitting (Stabbing) connected objects by axis-parallel hyperplanes:** Given a finite set of axis-parallel hyperplanes $\mathcal{P} \subseteq \mathbb{R}^d$ and a finite collection $\mathcal{Q}$ of connected objects in $\mathbb{R}^d$, find all minimal sets of hyperplanes from $\mathcal{P}$ that stab every object in $\mathcal{Q}$.

We show that the above two problems can be solved in incremental polynomial time, if the dimension d of the underlying space is fixed.

In the next section, we give a general framework for finding all minimal hitting sets for a given hypergraph. In Sections 5.2.2 and 5.2.3 we apply this framework to the problem of hitting hyper-rectangles by points and stabbing connected objects by axis-parallel hyperplanes respectively.

5.2.1 A Framework for Computing Transversal Hypergraphs

Let $\mathcal{H}$ be a hypergraph on a set $V = \{1, \ldots, n\}$. Recall the incremental approach of Section 2.4.3 that divides the problem of computing $\mathcal{H}^d$ into several sub-problems of the same type. The key fact behind the framework presented in this section is that for some geometric hypergraphs if we recurse on these sub-problems then we make some progress so that the process eventually terminates.

Given an ordering σ of vertices and for $i = 1, \ldots, n$, let $\mathcal{H}_i$ be the sub-hypergraphs of $\mathcal{H}$ defined as $\mathcal{H}_i = \{H \in \mathcal{H} : \sigma(i) \in H \text{ and } H \subseteq \{\sigma(1), \ldots, \sigma(i)\}\}$ and let $\Delta_i[X] = \{H \in \mathcal{H}_i : H \cap X = \emptyset\}$. A framework based on incremental approach of Section 2.4.3 is presented as Algorithm 7. The algorithm proceeds inductively, for $i = 1, \ldots, n$, by extending each minimal transversal X in $(\mathcal{H}_1 \cup \ldots \cup \mathcal{H}_{i-1})^d$ to a set in $(\mathcal{H}_1 \cup \ldots \cup \mathcal{H}_i)^d$ by recursively finding $(\Delta_i[X])^d$, each set of which is combined with X to obtain a minimal transversal of $(\mathcal{H}_1 \cup \ldots \cup \mathcal{H}_i)$.

Algorithm 7 A framework to compute minimal transversals of geometric hypergraph

```
Procedure Dualize1($\mathcal{H}, \Sigma, j$):
Input: A hypergraph $\mathcal{H}$ over $V = \{1, \ldots, n\}$, an index $j\ (\leq k)$ and a sequence $\Sigma = (\sigma_1, \ldots, \sigma_k)$ of permutations of vertices in $V$.
Output: The hypergraph $\mathcal{H}^d$.
1: $\mathcal{H}_0 \leftarrow \emptyset,\ \mathcal{X}_0 \leftarrow \{\emptyset\}, \mathcal{X}_i \leftarrow \emptyset\ \forall i = 1, \ldots, n$
2: for $i = 1, \ldots, n$ do
3:    Let $\mathcal{H}_i \leftarrow \{H \in \mathcal{H} : \sigma_j(i) \in H$ and $H \subseteq \{\sigma_j(1), \ldots, \sigma_j(i)\}\}$
4:    for all $X \in \mathcal{X}_{i-1}$ do
5:       Let $\Delta_i[X] = \{H \in \mathcal{H}_i : H \cap X = \emptyset\}$
6:       if $j = k$ or $|\Delta_i[X]| \leq 1$ then
7:          $\mathcal{A} \leftarrow$ Dualize-Simple($\Delta_i[X]$)
8:       else
9:          $\mathcal{A} \leftarrow$ Dualize1($\Delta_i[X], \Sigma, j+1$)
10:         $\mathcal{X}_i \leftarrow \min(\mathcal{X}_i \bigcup \{X \cup Y : Y \in \mathcal{A}\})$
11: return $\mathcal{X}_n$
```

The algorithm uses a sequence of permutations $\Sigma = \sigma_1 \ldots \sigma_k$ as a part of the input. When called initially as **Dualize1**$(\mathcal{H}, \Sigma, 1)$, it dualizes $\mathcal{H}$ by using the above mentioned approach where σ_j is used for partitioning in the j-th level of the recursion. After k levels of recursion, procedure **Dualize-Simple**() is used directly to solve the problem. As we will see in the later sections, for several classes of geometric hypergraphs, the subproblem after k levels can be solved easily, where k depends only on the dimension of the geometric space under consideration.

As an illustration, consider the problem of dualizing an *interval* hypergraph: Let $V = \{p_1, p_2, \ldots, p_n\}$ be a set of points on the line ordered from left to right, and let $\mathcal{H} \subseteq 2^V$ be a Sperner hypergraph, in which each edge $H \in \mathcal{H}$ consists of consecutive points from V. Denote by σ the left-to-right ordering of the vertices, and consider the execution of the Algorithm 7 when called as **Dualize1**$(\mathcal{H}, \Sigma, 1)$ with $\Sigma = \sigma$. The algorithm incrementally dualizes the hypergraphs $\mathcal{H}_i = \{H \in \mathcal{H} : p_i \in H$ and $H \subseteq \{p_1, \ldots, p_i\}\}$ for $i = 1 \ldots n$. Note that the subproblem $\Delta_i = \mathcal{H}_i$ contains at most one edge because of our assumption that $\mathcal{H}$ is Sperner and thus can be solved trivially.

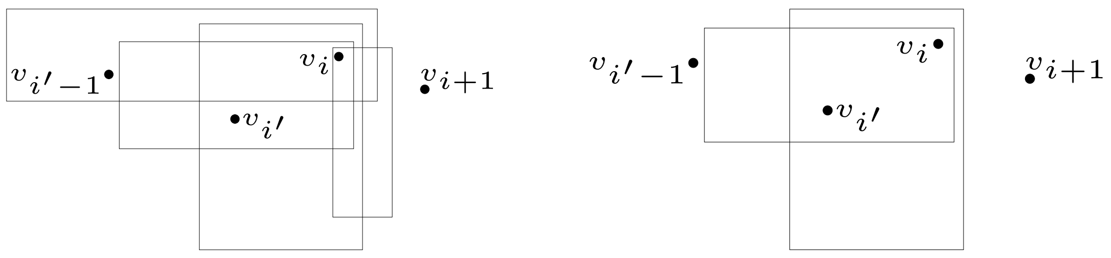

Figure 5.1: An example of points and rectangles in $\mathbb{R}^2$. Left: The set Δ_i consists of all rectangles that contain v_i and other points only from the subset $\{v_1, \ldots, v_i\}$. Right: The subproblem in the recursive call considers all rectangles which contain both v_i and $v_{i'}$ and no points from the strict left of $v_{i'}$ nor from the strict right of v_i

The correctness of the procedure follows from Proposition 2.10 and the correctness of the procedure **Dualize-Simple**. From Lemma 2.15 and the fact that Algorithm 7 recurses until depth at most k, we get the following bound on the worst-case running time.

Theorem 5.1. *Let $\mathcal{H} \subseteq 2^V$ be a hypergraph over vertex set V and $\Sigma = (\sigma_1, \ldots, \sigma_k)$ be a sequence of permutation functions of vertices of $\mathcal{H}$. Then procedure* **Dualize1**$(\mathcal{H}, \Sigma, 1)$ *computes $\mathcal{H}^d$ in $O((nm')^k(mm' + T))$ time, where $n = |V|$, $m = |\mathcal{H}|$, $m' = |\mathcal{H}^d|$ and T is time required by* **Dualize-Simple**$()$ *in each k-th level recursion of the procedure.*

5.2.2 Points and Hyper-rectangles in $\mathbb{R}^d$

Let $\mathcal{P}$ be a set of points and $\mathcal{Q}$ be a collection of axis-parallel hyper-rectangles in $\mathbb{R}^d$. In this section, we consider the problem of enumerating all minimal hitting sets of the hypergraph $(\mathcal{P}, \mathcal{Q})$. Let $\mathcal{H} \subseteq 2^{\mathcal{P}}$ be the hypergraph $(\mathcal{P}, \mathcal{Q})$.

To illustrate the idea, let us first consider the problem in $\mathbb{R}^2$. The algorithm is based on the framework presented as Algorithm 7. We order the points in $\mathcal{P}$ from *left* to *right* and if their x-coordinates are equal, we sort them from *bottom* to *top*. Let $v_1, v_2, \ldots, v_n$ be the corresponding ordering of the vertices of the hypergraph $\mathcal{H} \subseteq 2^{\mathcal{P}}$, defined above. Note that because of our ordering of the vertices, no rectangle in the hypergraph $\mathcal{H}_i$ contains any point strictly to the right of v_i and by definition, every rectangle in $\Delta_i \subseteq \mathcal{H}_i$ contains v_i.

Consider the subproblem of dualizing $\Delta_i[X]$ for each $i \in [n]$ and $X \in (\mathcal{H}_{i-1})^d$ and let the primed variables denote the corresponding variables in the recursive call of the algorithm. We order the vertices of $\mathcal{H}' = \Delta_i[X]$ in the *reverse* order i. e., from *right* to *left*, breaking ties by sorting them from *top* to *bottom*.

Consider now a further recursive call of the procedure on a hypergraph $\mathcal{H}'' = \Delta'_{i'}[X']$, where $i' \in \{1, \dots, |V(\mathcal{H}')|\}$ and $X' \in (\mathcal{H}'_{i'-1})^d$. The crucial observation is that each rectangle corresponding to an edge in the hypergraph $\Delta'_{i'}[X']$ contains both the points v_i and $v_{i'}$, and because of our ordering, no rectangle in the subproblem contains a point from the left of $v_{i'}$ nor from the right of v_i (see Figure 5.1). Hence, as can be easily seen, only the y-coordinates matter when deciding whether a given point $v \in \{v_{i'} \dots v_i\}$ intersects a rectangle from $\Delta'_{i'}[X']$. So we can project the subproblem $\Delta'_{i'}[X']$ on the y-axis and reduce it to a problem of dualizing an interval hypergraph, which can be solved in polynomial time, as seen in Section 5.2.1.

The above algorithm can be extended to higher dimensions in an obvious manner. In dimension d, we use the orderings $\Sigma = (\sigma_1, \dots, \sigma_{2d-2})$, where σ_i is the *lexicographical ordering* of the points using their last $d - \lfloor \frac{i-1}{2} \rfloor$ coordinates. Moreover, we define the ordering σ_i to be *increasing* when i is odd and *decreasing* otherwise. To generate all minimal transversals of $\mathcal{H}$, we call **Dualize1**$(\mathcal{H}, \Sigma, 1)$, and use the dualization procedure for interval hypergraphs in place of **Dualize-Simple**$()$. After the second recursive call the subproblems we obtain contain all hyper-rectangles that intersect two given points, say $v_{i'}$ and v_i with $v_{i'}$ being lexicographically smaller than v_i, and have the property that they contain no point that is lexicographically smaller then $v_{i'}$ or lexicographically greater than v_i. Hence after two levels of recursion, the first coordinate of the points can be ignored and thus the problem reduces to $d-1$-dimensional subproblems.

5.2.3 Stabbing Connected Objects in $\mathbb{R}^d$

Given a collection of connected objects $\mathcal{Q}$ and a set of axis-parallel hyperplanes $\mathcal{P}$, both in $\mathbb{R}^d$, we are interested in the problem of finding all minimal sets of hyperplanes from

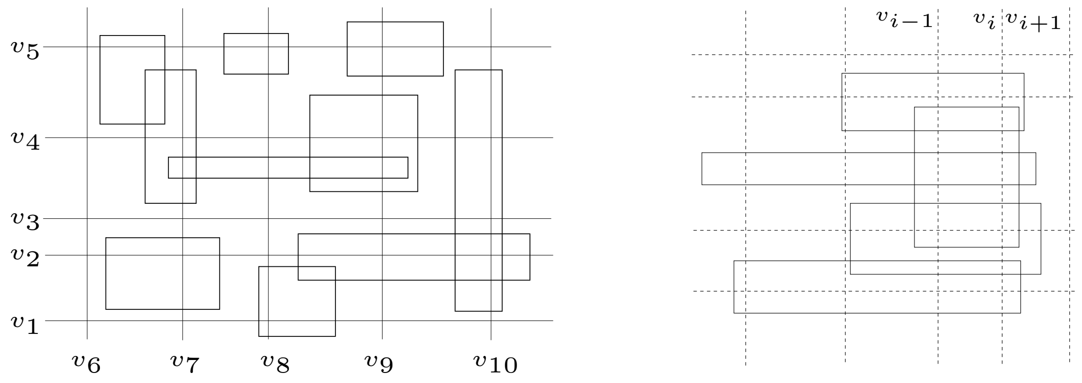

Figure 5.2: Left: An example of lines and objects in $\mathbb{R}^2$ and a valid ordering of lines. Right: The set $\Delta_i[X]$ consists of new objects that intersect v_i and only lines from the subset $\{v_1, \ldots, v_{i-1}\}$.

$\mathcal{P}$ such that every object in $\mathcal{Q}$ is stabbed by at least one of the hyperplanes in the set. Let $\mathcal{H} = (\mathcal{P}, \mathcal{Q})$ be the corresponding hypergraph with vertex set $\mathcal{P}$ and each object $Q \in \mathcal{Q}$ defining an edge consisting of all hyperplanes from $\mathcal{P}$ which intersect Q.

We consider the simple case first, that is, when all hyperplanes in $\mathcal{P}$ are parallel to each other. This turns out to be equivalent to the interval hypergraph case since we can project the objects in $\mathcal{Q}$ on any line L that is perpendicular to the hyperplanes in $\mathcal{P}$. The projection maps hyperplanes in $\mathcal{P}$ into points on L, and because of the connectedness, the objects in $\mathcal{Q}$ are mapped to intervals on L.

More generally, for $d > 1$, assume there is at least one hyperplane perpendicular to every principal axis. The assumption can be made without loss of generality, since if there is no hyperplane along a particular principal axis, say z, then we can orthogonally project all other hyperplanes and objects on the hyperplane $z = 0$ and reduce the dimension of the problem by one.

The dual of $\mathcal{H}$ can be found incrementally by following Algorithm 7. Fixing the order of principal axes, we order the hyperplanes sequentially: starting with hyperplanes perpendicular to the first principal axis, sorted in increasing order, we continue with the hyperplanes perpendicular to the second principal axis and so on. For an example in $\mathbb{R}^2$, the set of lines $\{x = 1, y = -1, x = 0, y = 1\}$ would be ordered as $x = 0, x = 1, y = -1, y = 1$ assuming that x-axis comes before y-axis in our fixed ordering of principal axes. Let σ be an ordering of hyperedges of $\mathcal{H}$ as defined above and let $j_0^{\mathcal{H}} < j_1^{\mathcal{H}} < \ldots < j_d^{\mathcal{H}}$ be indices with $j_0^{\mathcal{H}} = 0$ and $j_d^{\mathcal{H}} = n$, such that the corresponding hyperplanes in the group $\sigma(j_{r-1}^{\mathcal{H}} + 1), \ldots, \sigma(j_r^{\mathcal{H}})$ are parallel to each other and perpendicular to r-th principal axis for $r \in [d]$.

Consider the subproblem of dualizing $\mathcal{H}_i$ for $i = 1, \ldots, n$ as defined in the algorithm, where $\mathcal{H}_i$ contains only those edges which form subsets of vertices from $v_{\sigma(1)}, \ldots, v_{\sigma(i)}$. As discussed above, the problem reduces to dualizing an interval hypergraph when $1 \leq i \leq j_1^{\mathcal{H}}$. Now consider the case when $j_{r-1}^{\mathcal{H}} < i \leq j_r^{\mathcal{H}}$ for $r \in [d]$. Consider the subproblem of dualizing $\mathcal{H}' = \Delta_i[X]$ for $X \in (\mathcal{H}_{i-1})^d$, and let the primed variables denote the corresponding variables in the recursive call of the algorithm. Note that the subproblem for $\Delta_i[X]$ contains all objects that do not intersect any hyperplane "above" $v_{\sigma(i)}$. Let σ' be an ordering of vertices of $\mathcal{H}'$ defined similarly as σ for the hyperplanes perpendicular to first $r-1$ principal axes except the r-th group of hyperplanes which are sorted in decreasing order. As before, let $j_0^{\mathcal{H}'} < \ldots < j_r^{\mathcal{H}'}$ be indices with $j_0^{\mathcal{H}'} = 0$ and $j_r^{\mathcal{H}'} = n'$, where $n' = |V(\mathcal{H}')|$ and the hyperplanes in the group $\sigma'(j_{r'-1}^{\mathcal{H}'} + 1), \ldots, \sigma'(j_{r'}^{\mathcal{H}'})$ are parallel to each other and perpendicular to r'-th principal axis for $r' \in [r]$.

In the recursive call, we use σ' as our ordering and dualize $\mathcal{H}'$ incrementally by considering $\mathcal{H}'_{i'}$ for $1 < i' \leq i$. Note that for $i' \leq j_{r-1}^{\mathcal{H}'}$, $V(\mathcal{H}'_{i'}) \subseteq V(\mathcal{H}_{i-1})$ and hence the subproblem $\mathcal{H}'_{i'}$ is already taken care of when $(\mathcal{H}_{i-1})^d$ is computed. Alternatively, when $j_{r-1}^{\mathcal{H}'} < i' \leq i$ then similar to the case in Section 5.2.2, the subproblems $\Delta'_{i'}$ we get contain all objects that intersect both v_i and $v_{i'}$ with the property that no hyperplane above v_i or below $v_{i'}$ stabs any of them. Note that both $\{v_i\}$ and $\{v_{i'}\}$ as well as all hyperplanes between them are trivially hitting sets for the subproblems for hypergraphs of the form $\Delta'_{i'}[\cdot]$. The other hitting sets can be found recursively by observing that they do not involve any of the hyperplanes parallel to v_i and $v_{i'}$. Thus we are able to reduce the dimension of the problem by 1 after two levels of recursion.

In summary, to generate $\mathcal{H}$, we call **Dualize1**$(\mathcal{H}, \Sigma, 1)$ with $\Sigma = (\sigma_1, \sigma_2, \ldots, \sigma_{2d-1})$ and a trivial dualization procedure for **Dualize-Simple**$()$. For odd r, $1 \leq r < 2d$, the ordering σ_r sorts each group of parallel hyperplanes in increasing order (of the points of intersection with the common orthogonal line), whereas for even r, $1 < r < 2d$, σ_r is defined by sorting each group of parallel hyperplanes in increasing order except the last group, which is sorted in decreasing order (of the points of intersection with the common orthogonal line). As we noted above, at every r-th level of recursion for even r, the dual hypergraphs $\mathcal{H}_i^d$ such that $v_{\sigma(i)}$ does not belong to the last group of hyperplanes, can be easily computed from the corresponding dual hypergraph in the recursion level $r-1$. This observation can be used to avoid redundant computations by solving those subproblems directly instead of following the Algorithm 7.

5.3 Second Approach: Using Cuttings and Simplicial Partitions

In this section, we show that, when the hypergraph $(\mathcal{P}, \mathcal{Q})$ admits a *balanced subdivision*, then a recursive decomposition can be used to obtain efficiently all minimal hitting sets of the hypergraph. We apply this decomposition framework to get incremental polynomial-time algorithms for finding minimal hitting sets and minimal set covers for a number of hypergraphs induced by a set of points and a set of geometric objects. The set of geometric objects includes half-spaces, hyper-rectangles and balls, in fixed dimension.

5.3.1 Introduction

As compared with previous section, the framework of this section is simpler and it works for both the hitting set and set covering versions, and extends to more general objects, such as balls, half-spaces, and polytopes with fixed number of facets. In fact, as we will see, all that is needed is that the hypergraph admits a certain *balanced subdivision* which can be shown to exist in several geometric instances. One more important property of the algorithms we obtain, is that they admit a *global parallel* implementation, in the sense that all minimal hitting sets can be generated in polylogarithmic time using a polynomial number of processors. Among all polynomially solvable classes of hypergraphs, only very few are known to exhibit this nice property, see [KBEG07a, KBGE07].

Our algorithm is based on recursive decompositions of the space, and builds on the *full cover decompositions*, introduced in Section 2.4.5. In the following, we show that a modified version of a full cover can be combined with simplicial partitions and cuttings, to define a large class of hypergraphs for which full cover decompositions are effective.

The rest of Section 5.3 is organized as follows: In Section 5.3.2, we describe the type of decomposition used in the algorithm and show how it can be achieved for various geometrically induced range spaces using simplicial partitions and cuttings. We present the algorithm and analyze it in Section 5.3.3. We conclude with an application of generating minimal hitting sets of hypergraphs to a problem in mining spatial databases.

5.3.2 Balanced Subdivisions

Given any range space $(\mathcal{P}, \mathcal{Q})$, we say that a subset $\mathcal{P}' \subseteq \mathcal{P}$ is *stabbed* by a range $Q \in \mathcal{Q}$ if there exist $x, y \in \mathcal{P}'$ s.t. $x \in Q$ and $y \notin Q$. A *balanced subdivision* for a range space $(\mathcal{P}, \mathcal{Q})$ is a collection of a constant number of subsets $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_\lambda$ of $\mathcal{P}$ such that

1. For each $i \in \{1, \ldots, \lambda\}$, $|\mathcal{P}_i| \geq \epsilon|\mathcal{P}|$ for some constant $0 < \epsilon \leq 1$.
2. For each range $Q \in \mathcal{Q}$, there are two disjoint subsets $\mathcal{P}_i$ and $\mathcal{P}_j$ in the collection which are not stabbed by Q.

Remark 5.2. *In the above definition, the fact that a subset $\mathcal{P}' \subset \mathcal{P}$ is not stabbed by $Q \in \mathcal{Q}$ implies that $\mathcal{P}'$ is also not stabbed by $\mathcal{P} \setminus Q$. Consequently, we conclude that a balanced subdivision for any range space $(\mathcal{P}, \mathcal{Q})$ is also a balanced subdivision for the range space $(\mathcal{P}, \mathcal{Q}^c)$,* *where $\mathcal{Q}^c$ denotes the compliments of ranges in $\mathcal{Q}$, i. e., $\mathcal{Q}^c = \{\mathbb{R}^d \setminus Q \mid Q \in \mathcal{Q}\}$.*

In the next section, we show that if we can compute a balanced subdivision for a range space then we can enumerate its minimal hitting sets in global parallel time.

In this section, we show that several geometrically induced range spaces admit balanced subdivisions which can be computed efficiently (in parallel). Specifically, we show that for any point set $P \subset \mathbb{R}^d$, a balanced subdivision exists and can be computed efficiently for both $(P, \mathcal{H})$ and $(\mathcal{H}, P)$ if $\mathcal{H}$ is a family of objects of the following kind: (i) Half-Spaces in $\mathbb{R}^d$ (ii) Polytopes with a constant number of facets in $\mathbb{R}^d$ and (iii) Balls in $\mathbb{R}^d$.

We will use the following results:

Theorem 5.3 (Fine Simplicial Partitions [Mat92])**.** *Given any set P of n points in $\mathbb{R}^d$ and any parameter $1 \leq r \leq n$, there exists a partition $\Pi = \{P_1, P_2, \ldots, P_t\}$ of $t \leq r$ disjoint subsets of P and a set $\Delta = \{\Delta_1, \ldots, \Delta_t\}$ of simplices with the following properties: (i) $P_i \subseteq \Delta_i$ (ii) $\bigcup_i P_i = P$, (iii) $n/r \leq |P_i| \leq 2n/r$ for all $i \in \{1, \ldots, t\}$ and (iv) no half-space in $\mathbb{R}^d$ intersects more than $C_d r^{1-1/d}$ of the simplices in Δ, where C_d is a constant for any fixed d. The last property also implies that no half-space in $\mathbb{R}^d$ stabs more than $C_d r^{1-1/d}$ of the sets in Π. Further, for any $\delta > 0$, such a Π can be computed in time $O(n^{1+\delta})$. When r is bounded by a constant, Π can be computed in $O(n)$ time.*

Theorem 5.4 (Cuttings [Cha93])**.** *Given any set of n half-spaces in $\mathbb{R}^d$ and any parameter $1 \leq r \leq n$, there exists a partition of $\mathbb{R}^d$ into r simplices such that none of the simplices is stabbed by more than $C'_d n/r^{1/d}$ of the given half-spaces. Further, for any $\delta > 0$, such a partition can be computed deterministically in time $O(nr^{1-1/d})$.*

Parallel Implementation: Even though we only mention sequential running times above, such fine simplicial partitions and cuttings can be computed in $\mathrm{polylog}(n)$ time using $\mathrm{poly}(n)$ processors. For example, in the case of cuttings in any fixed dimension d, while the simplices are allowed to be arbitrary, it can be argued that they can always be chosen so that they are among a polynomial number of *canonical* simplices. In fact, it is not hard to argue that we can restrict to simplices whose corners are defined by the intersection of d of the hyperplanes defining the given set of halfspaces and indeed the construction in [Cha93] does restrict to such simplices. The number of such simplices is at most $O(n^{d(d+1)})$. Once we have a polynomial bound on the number of canonical simplices, we can check all possible sets of t canonical simplices in parallel using a polynomial number of processors and find a simplicial partition in polylogarithmic time. A similar argument holds for simplicial partitions.

Half-Spaces. Let us first consider the case when $\mathcal{H}$ is a set of half-spaces in $\mathbb{R}^d$. Given any set P with n points in $\mathbb{R}^d$, we can set r to be a large enough constant so that $C_d r^{1-1/d} \leq r - 2$. Then, clearly, the collection Π given by Theorem 5.3 also gives us a balanced subdivision of $(P, \mathcal{H})$ with $\epsilon = 1/r$ and $k \leq r$. To get a balanced subdivision of $(\mathcal{H}, P)$, we apply Theorem 5.4 to the half-spaces in $\mathcal{H}$. Assuming that $|\mathcal{H}| = n$, we set r to be a large enough constant so that $C'_d n/r^{1/d} \leq n/2$. Theorem 5.4 gives a partition of $\mathbb{R}^d$ into r regions $R_1, \ldots, R_r$ each of which is stabbed by at most $n/2$ half-spaces in $\mathcal{H}$. Consequently, for each region R_i, we either have at least $n/4$ half-spaces of $\mathcal{H}$ each of which contains R_i or we have at least $n/4$ half-spaces of $\mathcal{H}$ none of which intersects R_i. Let $\mathcal{H}_i$ be a set of those half-spaces. Then $|\mathcal{H}_i| \geq n/4$. We arbitrarily partition each $\mathcal{H}_i$ into two disjoint sets $\mathcal{H}_i^1$ and $\mathcal{H}_i^2$ each of size at least $n/8$. These sets $\mathcal{H}_i^1$ and $\mathcal{H}_i^2$ for $i \in \{1, \ldots, r\}$ give us a balanced subdivision of $(\mathcal{H}, P)$ with $k = 2r$ and $\epsilon = 1/8$ since each point $p \in P$ lies in some region R_j and hence does not stab the disjoint sets $\mathcal{H}_j^1$ and $\mathcal{H}_j^2$.

Remark 5.5. *In the case of half-spaces, one may also reduce the problem of finding minimal set covers to that of finding minimal hitting sets by using* geometric *duality, which maps points in $\mathbb{R}^d$ to hyperplanes and vice versa (see e.g. [dBvKOS97], Chapter 8). However this method does not work for polytopes.*

Polytopes. Suppose now that $\mathcal{H}$ is a set of polytopes in $\mathbb{R}^d$, each with at most f facets. In this case the collection Π given by Theorem 5.3 with r being a large enough constant so that $f \cdot C_d r^{1-1/d} \leq r-2$ gives us the required balanced subdivision for $(P, \mathcal{H})$. This is because each facet of a polytope can stab at most $C_d r^{1-1/d}$ members of Π and therefore a polytope with at most f facets can stab at most $f \cdot C_d r^{1-1/d} \leq r-2$ members of Π. To get a balanced subdivision for $(\mathcal{H}, P)$, we consider for each $H \in \mathcal{H}$, the set of at most f half-spaces whose intersection forms H. Let $\mathcal{H}'$ be the set of all these half-spaces. Assuming that $|\mathcal{H}| = n$, we have that $|\mathcal{H}'| \leq fn$. We then invoke Theorem 5.4 for the half-spaces in $\mathcal{H}'$ with r being a large enough constant so that $C'_d(nf)/r^{1/d} \leq n/2$. The regions in the resulting partition are stabbed by at most $n/2$ half-spaces in $\mathcal{H}'$ and hence at most $n/2$ polytopes in $\mathcal{H}$. We can then construct the balanced subdivision for $(\mathcal{H}, P)$ consisting of sets $\mathcal{H}_i^1$ and $\mathcal{H}_i^2$, $i \in \{1, \ldots, r\}$, as in the last paragraph.

Balls. Finally assume that $\mathcal{H}$ is a set of balls in $\mathbb{R}^d$. There is a standard *lifting* (Veronese map) which maps each point in $\mathbb{R}^d$ to a point in $\mathbb{R}^{d+1}$ and each ball in $\mathbb{R}^d$ to a half-space in $\mathbb{R}^{d+1}$ so that the incidence relations among them are preserved. Since balanced subdivisions exist for half-spaces in $\mathbb{R}^{d+1}$, we can conclude that balanced subdivisions exist for balls in $\mathbb{R}^d$ as well (for both $(P, \mathcal{H})$ and $(\mathcal{H}, P)$).

Since we invoke Theorems 5.3 and 5.4 with r being a constant, the collection in Theorem 5.3 and the partition in Theorem 5.4 can be computed in time $O(n)$. It follows that balanced subdivisions for the above range spaces can be computed in $O(n+m)$ time where n is the size of the ground set and m is the number of ranges.

5.3.3 The Enumeration Algorithm

Given a range space $(\mathcal{P}, \mathcal{Q})$, a divide-and-conquer algorithm to enumerate all minimal hitting sets of $\mathcal{Q}$ is presented as Algorithm 8. If $|\mathcal{P}|$ is at most some fixed constant μ, we use the procedure **Dualize-Simple** from Section 2.4.1.1 for the enumeration of minimal hitting sets in these trivial cases.

When $|\mathcal{P}| > \mu$, we assume the existence of a balanced subdivision $\Pi = (\mathcal{P}_1, \ldots, \mathcal{P}_\lambda)$, where λ is a constant and for each $i \in \{1, \ldots, \lambda\}$, $|\mathcal{P}_i| \leq \epsilon|\mathcal{P}|$ where $0 < \epsilon < 1$ is another constant. We classify the minimal hitting sets of $\mathcal{Q}$ into two types. Type 1

Algorithm 8 Procedure Dualize($\mathcal{P}$, $\mathcal{Q}$):

```
Input: A finite range space ($\mathcal{P}$, $\mathcal{Q}$)
Output: The set of all minimal hitting sets of $\mathcal{Q}$
 1: if $|\mathcal{P}| \leq \mu$ then
 2:    return Dualize-Simple($\mathcal{P}$, $\mathcal{Q}$)
 3: Type1-Set:=$\emptyset$
 4: Compute a balanced subdivision $\mathcal{P}_1, \ldots, \mathcal{P}_\lambda$ of ($\mathcal{P}$, $\mathcal{Q}$)
 5: for $i = 1, \ldots, \lambda$ do
 6:    Type1-Set := Type1-Set $\cup$ Dualize($\mathcal{P} \setminus \mathcal{P}_i$, $\mathcal{Q}^{\mathcal{P} \setminus \mathcal{P}_i}$)
 7: Type1-Set := Remove-Duplicates(Type1-Set)
 8: Type2-Set := $\emptyset$
 9: for $i = 1, \ldots, \lambda$ do
10:    $\mathcal{X}_i$ := Enumerate($\mathcal{P} \setminus \mathcal{P}_i$, $\mathcal{Q}_{\mathcal{P} \setminus \mathcal{P}_i}$)
11: for each $(M_1, \ldots, M_\lambda) \in \mathcal{X}_1 \times \ldots \times \mathcal{X}_\lambda$ do
12:    $M$ := $\bigcup_i M_i$
13:    if $M$ is a type 2 minimal hitting set of $\mathcal{Q}$ then
14:       Type2-Set := Type2-Set $\cup$ $\{M\}$
15: Type2-Set := Remove-Duplicates(Type2-Set)
16: return Type1-Set $\cup$ Type2-Set
```

minimal hitting sets are those that have an empty intersection with one of the $\mathcal{P}_i$s. The remaining minimal hitting sets which contain elements from each $\mathcal{P}_i$ are of type 2. Note that by Corollary 2.13, $(\mathcal{Q}^d)_{\mathcal{P} \setminus \mathcal{P}_i} = (\mathcal{Q}^{\mathcal{P} \setminus \mathcal{P}_i})^d$, and so type 1 hitting sets are easily enumerated recursively. This is done in line 6 of Algorithm 8. Enumerating Type 2 minimal hitting sets requires more work.

Let us first recall that any minimal hitting set M of $\mathcal{Q}$ and for any $v \in M$, there is always some range $Q \in \mathcal{Q}$ which *requires* v, i.e., $Q \cap M = \{v\}$. We call such a range a *certificate* range for v in M. Clearly, M is also a minimal hitting set for the set of certificate ranges of its elements.

Let M be any type 2 minimal hitting set and let $Q \in \mathcal{Q}$ be any range that has a nonempty intersection with each of the $\mathcal{P}_i$s. Since Π is a balanced subdivision, there are at least two sets $\mathcal{P}_j$ and $\mathcal{P}_k$ which are not stabbed by Q. Since Q has a nonempty intersection with both of them, it must contain both the sets as subsets. Now, since M contains an element from each $\mathcal{P}_i$, Q contains at least two elements of M implying that Q cannot be a certificate range for any element of M. This means that for the purpose of enumerating type 2 minimal hitting sets, we can discard all ranges which have a non-empty intersection with each of the $\mathcal{P}_i$s. Let $\mathcal{Q}_i = \mathcal{Q}_{\mathcal{P}\setminus\mathcal{P}_i}$ and let $\tilde{\mathcal{Q}} = \bigcup_i \mathcal{Q}_i$.

Let M be any type 2 minimal hitting set of $\mathcal{Q}$. Since $\tilde{\mathcal{Q}}$ contains all certificate ranges of M, M is also a minimal hitting set for $\tilde{\mathcal{Q}}$. Also, since the ranges in $\mathcal{Q}_i$ do not contain any element of $\mathcal{P}_i$, $M \setminus \mathcal{P}_i$ is a hitting set (not necessarily minimal) for $\mathcal{Q}_i$ and therefore contains some $M_i \subseteq M \setminus \mathcal{P}_i$ which is a minimal hitting set for $\mathcal{Q}_i$.

Note that each element of M appears in at least one of the M_is, since each $v \in M$ has a certificate range Q which belongs to some $\mathcal{Q}_i$ implying that $v \in M_i$. In other words, $M = \bigcup_i M_i$. This suggests the following algorithm for enumerating the minimal hitting sets of type 2. For each $i \in \{1, \ldots, \lambda\}$, recursively compute the set $\mathcal{X}_i$ of all minimal hitting sets of $(V \setminus V_i, \mathcal{Q}_i)$. Then, try all possible ways of picking one minimal hitting set $M_i \in \mathcal{X}_i$ from each $\mathcal{X}_i$ and output $M = \bigcup_i M_i$ if it is a type 2 minimal hitting set for $\mathcal{Q}$. This way we surely enumerate all type 2 minimal hitting sets. Now, we need to bound the number of combinations we try. We get this from Proposition 2.12 which implies that $|\mathcal{X}_i| \leq |\mathcal{Q}^d|$.

Let T be the number of minimal hitting sets of $\mathcal{Q}$. It follows that the number of combinations of M_i's we need to try is at most T^λ. After we find all type 1 minimal hitting sets we run a procedure called **Remove-Duplicates** to remove any duplicates we may have generated. Similarly, after we find all type 2 minimal hitting sets, we run **Remove-Duplicates** to remove any duplicates. This ensures that in the end we do not output any duplicates.

We now do an analysis of the running time of the algorithm. In the analysis, we treat the number of ranges $m = |\mathcal{Q}|$ and the number T of minimal hitting sets of $\mathcal{Q}$ as fixed. We denote by $t(n)$, the running time of the procedure **Dualize** on a hypergraph $(\mathcal{P}, \mathcal{Q})$ where $|\mathcal{P}| = n$. The recursive calls in Line 6 of Algorithm 8 for enumerating type 1 minimal hitting sets take time $\lambda t((1-\epsilon)n)$. Similarly, the total time spent in Line 10 is $\lambda t((1-\epsilon)n)$. The loop starting on Line 11 is executed at most T^λ times. In each iteration, checking whether M is a type 2 minimal hitting set of $\mathcal{Q}$ takes $O(mn)$ time. Hence the total time spent in the loop is $O(mnT^\lambda)$. Since there are at most T distinct minimal hitting sets of $\mathcal{Q}$, when we reach Line 7, Type1-Set has at most λT minimal hitting sets. Each of these have to be tested against a set of at most T distinct minimal hitting sets to see if it has already been reported. Therefore, this takes $O(\lambda T^2 n)$ time assuming that it takes $O(n)$ to check if two minimal hitting sets are the same. Similarly, when we reach Line 15, the size of Type2-Set is at most T^λ and each of the hitting sets in it is compared against a set of at most T minimal hitting sets to see if it has been reported before. This takes $O(T^{\lambda+1}n)$ time. We therefore have the following recursion:

$$t(n) \leq 2\lambda t((1-\epsilon)n) + \lambda n T^2 + mnT^\lambda + nT^{\lambda+1} + \tau,$$

where τ is the time required to find a balanced subdivision. Using the fact that $t(n)$ is a constant when n is smaller than some constant μ, we see that $t(n) = O((\tau + nT^{\lambda+1} + nmT^\lambda) \cdot n^{\frac{\log \lambda}{\log 1/(1-\epsilon)}})$. We thus have the following theorem.

Theorem 5.6. *Procedure* **Dualize**$(\mathcal{P}, \mathcal{Q})$ *finds all minimal hitting sets of a range space* $(\mathcal{P}, \mathcal{Q})$ *which admits a balanced subdivision* $V_1, \ldots, V_\lambda$ *with each* $|V_i| \geq \epsilon|V|$*, whenever* $|V|$ *is larger than a fixed constant* μ*, in time* $O((\tau + nT^{\lambda+1} + nmT^\lambda) \cdot n^{\frac{\log \lambda}{\log 1/(1-\epsilon)}})$*, where* $n = |V|$*,* $m = |\mathcal{Q}|$*,* T *is the number of minimal hitting sets of* $\mathcal{Q}$ *and* τ *is the time required to compute a balanced subdivision.*

Remark 5.7. *The way the above algorithm is described gives an output polynomial algorithm for generating* $\mathcal{Q}^d$*. Using the techniques described in Section 2.4.5, we can modify the algorithm to become incremental polynomial, that is, for every* $T' \leq T$ *the algorithm outputs* T' *transversals in time polynomial in* n*,* m *and* T'*.*

Parallel Implementation of the Algorithm: Algorithm 8 can be parallelized in an obvious way. Each of the For loops can be executed in parallel, i.e., all the iterations are done in parallel. Using $\mathrm{poly}(n, m, T)$ processors, each of the other steps can be executed in $\mathrm{polylog}(n, m, T)$ time. If we denote by $t^{\|}(n)$ the running time of such a parallel algorithm, again treating m and T as constants, we get the following recurrence:

$t^{\|}(n) = t^{\|}((1-\epsilon)n) + \mathrm{polylog}(n, m, T)$. We therefore have that $t^{\|}(n)$ is in $\mathrm{polylog}(n, m, T)$. It can be checked that the total number of processors required is only $\mathrm{poly}\,(n, m, T)$.

As discussed in Section 2.4.5, we can also get an incremental version of this, i.e., for any $T' \leq T$, the running time depends polylogarithmically on M', provided that there is an efficient parallel algorithm for finding a single minimal transversal of the input hypergraph $\mathcal{Q}$ which is an outstanding open problem (see e.g. [KUW88]).

5.3.4 Application - Infrequent pointsets

Let $(\mathcal{P}, \mathcal{Q})$ be a range space induced by a set of points $\mathcal{P}$ and a set of geometric objects in $\mathbb{R}^d$. Recall from Section 2.5.2 that a set of points $X \subseteq \mathcal{P}$ is said to be *t-frequent*, for an integer t, if it is contained in at least t ranges in $\mathcal{Q}$ and is *t-infrequent* otherwise.

Denote respectively by $\mathcal{F}_{\mathcal{Q},t}$ and $\mathcal{G}_{\mathcal{Q},t}$ the families of *minimal* t-infrequent and *maximal* t-frequent pointsets with respect to $\mathcal{Q}$. The generation of minimal infrequent and maximal frequent sets is an important task in data mining applications, see e.g. [AIS93, AMS+96, BGKM02]. In the case when the database stores geometrical information obtained from images or geographical data (the so-called *spatial databases*, see e.g. [MPV05]), the mining process may involve finding frequent or infrequent pointsets with respect to a given set of ranges.

Assume that the dimension d is fixed. For a class $\mathbb{Q}$ of ranges, denote by $\mathbb{Q}_\cap$ the class of all possible intersections of ranges in $\mathbb{Q}$.

Proposition 5.8. *Consider a class of ranges $\mathbb{Q}$, such that $\mathcal{Q}^d$ can be generated in polynomial time for any $\mathcal{Q} \in \mathbb{Q}_\cap^c$. Then for any t, all minimal t-infrequent pointsets with respect to $(\mathcal{P}, \mathcal{Q})$ can be generated in polynomial time.*

Proof. It is known [BGKM02] that the generation of $\mathcal{F}_{\mathcal{Q},t}$ reduces in polynomial time to checking whether two given families $\mathcal{X} \subseteq \mathcal{F}_{\mathcal{Q},t}$ and $\mathcal{Y} \subseteq \mathcal{G}_{\mathcal{Q},t}^c$ are dual to each other, i.e, $\mathcal{Y}^d = \mathcal{X}$. Since each set in $\mathcal{G}_{\mathcal{Q},t}$ can be identified with the intersection of all ranges containing it, $\mathcal{Y}$ can be regarded as a range space from the class $\mathbb{Q}_\cap^c$, and thus the transversal problem can be solved in polynomial time. □

For instance, the class of axis-parallel hyper-rectangles is closed under intersection, and hence by Remark 5.2, t-infrequent pointsets with respect to this class can be found in polynomial time.

5.4 Enumerating Minimal Hitting Sets of Half-Planes with Polynomial Delay

Although the algorithm of Section 5.3 covers half-planes in $\mathbb{R}^2$ as a special case, its running time is super-linear in the output size and hence not ideal for some applications. Similarly, the only other algorithm for generating minimal hitting sets of half-planes [EMR09], which is based on the framework presented in Section 5.2, suffers from the same short-coming. In this section we describe another algorithm for half-planes that produces output with polynomial delay, and hence, the total time to generate all minimal hitting sets is linear in the output size. Specifically, we show that the costly reduction employed by the algorithm of [EMR09] to subproblems of a special type, can be replaced by a much simpler and efficient computation.

Given a set of points P and a set of half-planes $\mathcal{R}$ (more precisely, subsets of P determined by half-planes), consider a minimal hitting set $M \subseteq P$ of $\mathcal{R}$. The minimality of M implies that, for every point $p \in P$, there is a half-plane $H_p \in \mathcal{R}$, (called a certificate for p in M,) such that $H_p \cap M = \{p\}$. Consequently, we conclude that all points of any minimal hitting set of $\mathcal{R}$ must lie in convex position. Assume w.l.o.g. that $|M| \geq 2$ and let $M' = \{p_1, p_2\} \subseteq M$ contain the top-most and bottom-most points of the convex hull of M. See Figure 5.3 for an illustration. Clearly, all the other points in M lie in one of the regions A_1 or A_2 in Figure 5.3. As a result, we can decompose the problem of finding all minimal hitting sets that contain M' with p_1 and p_2 as the top-most and bottom-most points of their convex hull into two independent subproblems. To

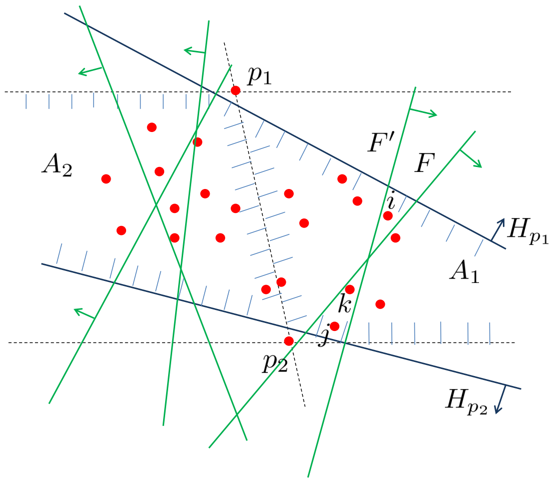

Figure 5.3: Points p_1 and p_2 are assumed respectively to be the top-most and bottom-most points in the convex hull of the minimal hitting set M. All other points in M lie the (shaded) regions A_1 and A_2.

describe such decomposition, we first need the following definition.

Recall from Proposition 2.8 that a non-empty subset $S \subseteq P$ is a sub-transversal for $\mathcal{R} \subseteq 2^P$ if and only if there is a non-blocked selection $\{H_p \in \mathcal{R}_p(S) \mid p \in S\}$ for S, where $\mathcal{R}_p(S) = \{H \in \mathcal{R} \mid H \cap S = \{p\}\}$ be the set of certificates for p.

Assuming that $M' = \{p_1, p_2\}$ is a sub-transversal, there is a non-blocked selection $\{H_{p_1}, H_{p_2}\}$, where H_{p_i} is a certificate for p_i. Define $P' = H_{p_1} \cup H_{p_2}$, and let $P_1, P_2 \subset P'$ be the sets of points in P' that lie in regions A_1 and A_2 in Figure 5.3, respectively. Finally, let $\mathcal{R}_i = \{H \setminus P' : \ H \in \mathcal{R}, \ H \cap M' = \emptyset, \ H \subset P_i\}$, for $i = 1, 2$. It is easy to see that any minimal transversal containing M', such that p_1 and p_2 are the top-most and bottom-most points of its convex hull, can be obtained as the union $M' \cup M_1 \cup M_2$, where $M_1 \in \mathcal{R}_1^d$ and $M_2 \in \mathcal{R}_2^d$, and where $\mathcal{R}_1$ and $\mathcal{R}_2$ are defined by some certificates H_{p_1} and H_{p_2}.

The advantage of such reduction is that $\mathcal{R}_1$ and $\mathcal{R}_2$ are both collections of *uni-directional* half-planes, in the following sense: there is a line ℓ (in Figure 5.3, ℓ is the line through p_1 and p_2) such that every half-plane in $\mathcal{R}_i$ has its normal pointing in the direction away form ℓ.

In the following, we show that we can use simple backtracking to dualize uni-directional half-planes with polynomial delay by exploiting a certain ordering on the points in P_1 and P_2. The next section described our backtracking procedure, while the criterion used to guide the backtracking is discussed in Section 5.4.2.

Algorithm 9 Procedure Enumerate-BT($\mathcal{R}, \sigma, i, S$):

```
Input: A hypergraph $\mathcal{R} \subseteq 2^V$, an ordering $\sigma$ on $V$, an integer $i \in \{1, \ldots, |V|\}$ and a
    subset $S \subseteq \sigma([i-1]) \overset{\text{def}}{=} \{\sigma(j) : j \in [i-1]\}$
Output: The set $\{M \in \mathcal{R}^d : M \supseteq S, M \cap (\sigma([i-1]) \setminus S) = \emptyset\}$
1: if $S \in \mathcal{R}^d$ then
2:     output $S$
3:     return
4: if $\exists M \in \mathcal{R}^d$ s.t. $S \cup \{\sigma(i)\} \subseteq M$ and $(\sigma([i-1]) \setminus S) \cap M = \emptyset$ then
5:     Enumerate-BT($\mathcal{R}, \sigma, i+1, S \cup \{\sigma(i)\}$)
6:     Enumerate-BT($\mathcal{R}, \sigma, i+1, S$)
```

5.4.1 Backtracking Method

The procedure which is similar to the backtracking algorithm presented in Section 2.4.2, is given as Algorithm 9. Since the algorithm essentially builds a backtracking tree whose leaves are the minimal transversals of $\mathcal{R}$, the time required to produce each new minimal transversal is bounded by the depth of the tree (at most $\min\{|V|, |\mathcal{R}|\}$) times the maximum time required at each node. The efficiency of such procedure depends on being able to perform the test in Step 4, which is addressed in the next subsection.

Without loss of generality we concentrate on finding $\mathcal{R}_1^d$, where $\mathcal{R}_1$ is the hypergraph consists of only uni-directional planes as defined above. The other set of transversals can be found similarly. In other words, we may assume that all points lie inside the region A_1 in Figure 5.3.

We use the backtracking method with σ being the following lexicographic order of the points: if $p = (p_1, p_2)$ and $q = (q_1, q_2)$ then $p \prec_\sigma q$ if and only if $p_1 < q_1$ or $p_1 = q_1$ and $p_2 < q_2$. Without loss of generality, we assume $V(\mathcal{R}_1) = \{1, \ldots, n\}$ and reorder the points such that they are numbered from 1 to n according to σ, i. e., assume that σ is the identity permutation.

5.4.2 Checking the Sub-Transversal Criterion

Now we show that the sub-transversal test in Step 4 of the backtracking procedure can be performed in polynomial time. Given $i \in [n]$ and $S \subseteq [i-1]$, we would like to check if

$$\exists M \in \mathcal{R}_1^d \text{ such that } S \cup \{i\} \subseteq M \text{ and } ([i-1] \setminus S) \cap M = \emptyset. \tag{5.1}$$

Lemma 5.9. *Fix $i \in [n]$ and let $S \subseteq [i-1]$ be a sub-transversal of $\mathcal{R}_1$. Suppose that $F, F' \in \mathcal{R}_1$ satisfy: $F \cap (S \cup \{i\}) = \{j\}$ for $j \neq i$ and $F' \cap (S \cup \{i\}) = \{i\}$. Then $F \setminus [i-1] \subseteq F' \setminus [i-1]$.*

Proof. We may assume without loss of generality that the picture looks as in Figure 5.3. If there is a point with index $k \in F \setminus ([i-1] \cup F')$ then k comes before i with respect to the first coordinate (see Figure 5.3), in contradiction to the fact that we process the points according to the order imposed by σ. □

Now the sub-transversal criterion of Proposition 2.8 reduces to a simple check.

Lemma 5.10. *Given $i \in [n]$ and $S \subseteq [i-1]$. Then* (5.1) *holds if and only if there exists $F \in \mathcal{R}_1$ such that $F \cap (S \cup \{i\}) = \{i\}$ and for all $F' \in \mathcal{R}_1$ for which $F' \cap (S \cup \{i\}) = \emptyset$, we have $(F' \setminus [i]) \setminus (F \setminus [i]) \neq \emptyset$.*

Proof. We apply the sub-transversal criterion for $S \cup \{i\}$ in the restricted hypergraph $\mathcal{R}'_1 = \{H \setminus ([i-1] \setminus S) : H \in \mathcal{R}_1\}$. By Proposition 2.8, (5.1) holds if and only if there exist $F_1, \ldots, F_i \in \mathcal{R}_1$ such that $F_j \cap (S \cup \{i\}) = \{j\}$, for $j = 1, \ldots, i$, and $(F' \setminus [i-1]) \not\subseteq \bigcup_{j=1}^{i}(F_j \setminus [i-1])$ for all $F' \in \mathcal{R}_1$ such that $F' \cap (S \cup \{i\}) = \emptyset$. By Lemma 5.9, the union $\bigcup_{j=1}^{i}(F_j \setminus [i-1])$ is equal to $F_i \setminus [i-1]$. □

By using geometric duality, which maps points in $\mathbb{R}^2$ to half-planes and vice versa (see e. g., [dBvKOS97], Chapter 8), we get the following theorem.

Theorem 5.11. *Let P be a set of n points and $\mathcal{R}$ be a set of m half-planes in $\mathbb{R}^2$. Then all minimal hitting sets of the range spaces $(P, \mathcal{R})$ and $(\mathcal{R}, P)$ can be generated in* $\operatorname{poly}(n, m) \cdot T$ *time where T is the size of the output.*

5.5 Conclusions

In this chapter, we presented two different approaches to dualize various geometric hypergraphs. We also gave a polynomial delay algorithm for the special case of hypergraph induced by half-planes and points in $\mathbb{R}^2$.

As a future research area, the efficient dualization of hypergraphs with bounded VC-dimension is a natural next step, which unifies different geometric hypergraphs we were able to efficiently dualize so far.

Chapter 6

Readability

Golumbic et al. [GMR06] defined the *readability* of a monotone Boolean function f to be the minimum integer k such that there exists an $\wedge$-$\vee$-formula equivalent to f in which each variable appears at most k times. They asked whether there exists a polynomial-time algorithm, which given a monotone Boolean function f, in CNF or DNF form, checks whether f is a read-k function, for a fixed k. In this chapter, we partially answer this question already for $k = 2$ by showing that it is NP-hard to decide if a given monotone formula represents a read-twice function. It follows also from our reduction that it is NP-hard to approximate the readability of a given monotone Boolean function $f : \{0,1\}^n \to \{0,1\}$ within a factor of $O(n)$. We also give tight sublinear upper bounds on the readability of a monotone Boolean function given in CNF (or DNF) form, parameterized by the number of terms in the CNF and the maximum size in each term, or more generally the maximum number of variables in the intersection of any constant number of terms. When the variables of the DNF can be ordered so that each term consists of a set of consecutive variables, we give much tighter logarithmic bounds on the readability.

6.1 Introduction

Let $f : \{0,1\}^n \to \{0,1\}$ be a monotone Boolean function, i. e., for any $x, x' \in \{0,1\}^n$, $x' \geq x$ implies $f(x') \geq f(x)$. One property of such functions is that they can be represented by negation-free Boolean formulae.

A monotone read-k formula is a Boolean formula over the operators $\{\vee, \wedge\}$ in which each variable occurs at most k times. The readability of f is the minimum k such that f can be represented by a monotone read-k formula. We also call f a read-k function when it has readability k. Finding the readability of an arbitrary Boolean function and computing a formula which achieves this readability has applications in circuit design among others and therefore is one of the earliest problems considered in Computer Science [GMR06].

Given a monotone Boolean function in one of the normal forms (CNF/DNF), a complete combinatorial characterization for it to be read-once was given by Gurvich [Gur77, Gur91]. A polynomial-time algorithm based on this criterion to decide whether a given CNF or DNF is read-once is given by Golumbic et al. [GMR06] . The algorithm also computes the unique read-once representation when a read-once function is given as input. For $k \geq 2$, no characterization is known for a given monotone Boolean CNF or DNF to be read-k, and in fact, Golumbic et al. asked in [GMR06] whether there exists a polynomial-time algorithm, which given a (normal) monotone Boolean function f in CNF or DNF form, checks whether f is a a read-k function, for a fixed k.

The case when the function is given by an oracle has also been considered in the machine learning community. It is shown in [AHK93] that given a read-once function by a membership oracle, we can compute its read-once representation in polynomial time. However, the correctness of the algorithm is based on the assumption that the function provided as an oracle is read-once. If it is not read-once then the algorithm terminates with incorrect output.

In this chapter, we show that, given an $\wedge$-$\vee$-formula, it is NP-hard to check if it represents a read-twice function f. In fact, we prove a stronger result: given a read-twice representation of f, it is hard to decide whether f is actually read-once. This partially answers the question of Golumbic et al. [GMR06], but leaves open the case when f is given by the CNF or DNF normal form. It follows also from our reduction that it is NP-hard to approximate the readability of a given monotone Boolean function $f : \{0,1\}^n \to \{0,1\}$ within a factor of $O(n)$.

It follows from a result in [Weg87] that almost all monotone Boolean functions on n variables, in which each minterm has size exactly k, have readability $\Omega(n^{k-1} \log^{-1} n)$. Assuming that the function is given by its irredundant DNF (or CNF) of m minterms, this implies a lower bound of $\tilde{\Omega}(m^{1-\frac{1}{k}})$ on the readability. This naturally raises the question whether this bound is tight, i.e. for any monotone CNF formula of m terms, there exists an equivalent read-$O(m^{1-\frac{1}{k}})$ representation. In this chapter, we show that this is indeed the case, and moreover that such a representation can be found in polynomial time. In fact, we prove a more general result. For integers $p, q > 0$, let us say a monotone CNF f has (p,q)-bounded intersection [KBEG07b] if every p terms intersect in at most q variables. We show that any such CNF has read-$O((p+q-1)m^{1-\frac{1}{q+1}})$ representation which can be found in polynomial-time. Confronted with this almost tight sublinear bound on readability, an interesting question is whether it can be improved for interesting special cases. For the class of interval DNF's, i.e. those for which there is an ordering on the variables such that each term contains only consecutive variables in that ordering, we show that readability is at most $4\lceil \log m \rceil$.

The rest of the chapter is organized as follows. In the next section, we point out that the characterization of [Gur77] for read-once functions does not carry over to read-k functions already for $k = 2$. In Section 6.3, we present upper bounds on the readability of some classes of monotone Boolean DNF (resp. CNF) that depend only on the number of terms in the normal form. In Section 6.4 we show that finding the readability in general is hard when the input formula is not a DNF or CNF. We also give an $O(n)$ inapproximability result in this case.

6.2 On Generalization of Read-Once Functions

An elegant characterization of read-once functions is provided by the following theorem of Gurvich.

Theorem 6.1 ([Gur77]). *For any monotone Boolean function f the following two statements are equivalent: (i) f is read-once. (ii) Every minterm and maxterm of f intersect in exactly $c = 1$ variable.*

However, this result does not generalize to read-twice functions as the following example shows. Consider the read-twice formula

$$g(x_1, \ldots, x_n, y_1, \ldots, y_n) = \bigwedge_{1 \le i \le n} (x_i \vee y_i) \bigwedge (x_1 \vee \ldots \vee x_n).$$

It is easy to see that g has a minterm $x_1 \ldots x_n$ which intersects with the maxterm $(x_1 \vee \ldots \vee x_n)$ in n variables. Hence hypergraphs corresponding to read-twice functions do not necessarily satisfy the generalization of condition (ii) of Theorem 6.1 for any constant $c > 1$. Conversely, any such generalization is also not sufficient for a function to be read-c, as implied by the following result on the shortest possible size of k-homogeneous DNF's where the size of each term is exactly k (and hence each minterm and maxterm intersect in at most k).

Theorem 6.2 (cf. [Weg87]). *For an integer k, let $\mathscr{H}_k^n$ be the class of monotone Boolean functions on n variables such that size of every minterm is exactly k. The monotone formula size of almost all $h \in \mathscr{H}_k^n$ is $\Omega(n^k \log^{-1} n)$.*

Theorem 6.2 implies that the readability of almost all $h \in \mathscr{H}_k^n$ is $\Omega(n^{k-1} \log^{-1} n)$, since otherwise the formula achieving a smaller readability has smaller than the shortest possible size.

6.3 Upper Bounds

In this section, we consider various classes of monotone Boolean DNF's and give upper bounds on their readability. First we consider Interval DNF's whose terms correspond to consecutive variables, given some ordering on the variables. Next, we consider (p, q)-intersecting DNF's where every p of its terms intersect in at most q variables and give an almost tight upper bound on their readability. Finally, we consider a special case of the latter class, namely k-DNF's, where the size of each term is bounded by k and again give a tight upper bound on their readability. Even though we get the same upper bound implied by the more general case, the formula computed by our algorithm has only depth 3 in this case.

In our description of the algorithms, we use set-theoretic notations to describe various operations on the structure of DNF's. In this sense, we treat the DNF $\phi = \bigvee t_i$ as its corresponding hypergraph $\{t_i \mid t_i \text{ is a term in } \phi\}$. For example, we write $t \in \phi$ when t is term of ϕ and similarly by $x \in t$ we mean that the term t contains variable x. Let us denote the *degree* of a variable in ϕ by $\deg_\phi(x)$, which is the number of terms in ϕ containing $x \in V$. For a Boolean formula f and a literal x (resp. set of literals S) in f, we denote by $f\mid_{x=1}$ (resp. $f\mid_{S=1}$) the resulting f after replacing every occurrence of x (resp. $x \in S$) in f with 1.

6.3.1 Interval DNF

A monotone Boolean DNF $\mathcal{I} = \bigvee_{I\in\mathcal{I}} \bigwedge_{i\in I} x_i$ is called interval DNF if there is an ordering of variables $V = \{x_1, x_2, \ldots, x_n\}$ such that each $I \in \mathcal{I}$ contains only consecutive elements from the ordering. We show that an interval DNF containing m terms is $O(\log m)$-readable. For a term $I = x_i x_{i+1} \ldots x_j$ in interval DNF $\mathcal{I}$, we call x_i and x_j its left and right end-points, and denote them with $\mathrm{L}(I)$ and $\mathrm{R}(I)$, respectively. We also denote the first (resp. last) term in the ordering of terms of $\mathcal{I}$ with respect to their left end point as $\mathrm{first}(\mathcal{I})$ and $\mathrm{last}(\mathcal{I})$, respectively.

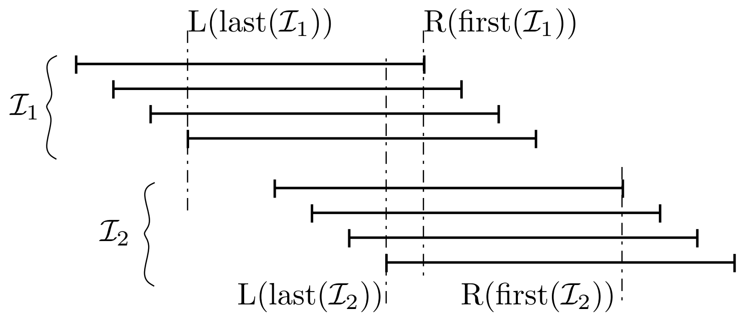

Figure 6.1: Proof of Theorem 6.3.

Let us call an interval DNF *intersecting* if all terms in it have a non-empty intersection. It is known that the terms of an irredundant interval DNF can be partitioned into two sets such that each set is a the disjoint union of intersecting DNF's(cf. [AJHL89]). Consequently, the readability of an irredundant interval DNF can be bounded by twice the maximum readability of an intersecting DNF in the partition. The algorithm to find a $2\lceil\log(m')\rceil$-readable formula for intersecting DNF $\mathcal{I}'$ consisting of m' terms is given as Algorithm 10. It first divides the terms in $\mathcal{I}'$ into two halves ($\mathcal{I}_1$ and $\mathcal{I}_2$) by considering them in the order of their left end-points. The common variables are then factored out from $\mathcal{I}_1$ and $\mathcal{I}_2$ and the equivalent formulae for the remaining parts are computed recursively.

Algorithm 10 An algorithm to find an $2\lceil\log(m)\rceil$-readable formula for interval DNF consisting of m intersecting terms

```
Procedure REDUCE1($\mathcal{I}$):
Input: A monotone Boolean interval DNF $\mathcal{I} = \bigvee_{j=1}^{m} I_j$ s.t. $\bigcap_{j=1}^{m} I_j \neq \emptyset$
Output: A $2\lceil\log(m)\rceil$-readable formula $\psi$ equivalent to $\mathcal{I}$
1: if $|\mathcal{I}| \leq 2$ then
2:     return the read-once formula representing $\mathcal{I}$
3: Consider terms in $\mathcal{I}$ in order of their left end points, let $\mathcal{I}_1$ (resp. $\mathcal{I}_2$) be first half
   (resp. remaining half) elements of $\mathcal{I}$.
4: Let $\phi_1$ (resp. $\phi_2$) be set of all variables that occur in every term of $\mathcal{I}_1$ (resp. $\mathcal{I}_2$)
5: $t_1 := \text{first}(\mathcal{I}_1),\ t_2 := \text{last}(\mathcal{I}_1),\ t_3 := \text{first}(\mathcal{I}_2),\ t_4 := \text{last}(\mathcal{I}_2)$
6: $\psi_1 = \text{REDUCE1}((\mathcal{I}_1 \setminus \{t_1, t_2\}) \mid_{\phi_1=1}), \quad \psi_2 = \text{REDUCE1}((\mathcal{I}_2 \setminus \{t_3, t_4\}) \mid_{\phi_2=1})$
7: return $(\phi_1 \wedge (\psi_1 \vee t_1 \mid_{\phi_1=1} \vee t_2 \mid_{\phi_1=1})) \bigvee (\phi_2 \wedge (\psi_2 \vee t_3 \mid_{\phi_2=1} \vee t_4 \mid_{\phi_2=1}))$
```

Theorem 6.3. *Let $\mathcal{I}$ be an irredundant interval DNF containing m terms. Then $\mathcal{I}$ is $4\lceil\log(m)\rceil$-readable.*

Proof. We first partition the terms of $\mathcal{I}$ into two sets such that each set is the union of disjoint intersecting DNF's. For each intersecting DNF $\mathcal{I}'$ in either of the classes, let $r(m')$ be the readability of the formulae generated by the procedure REDUCE1($\mathcal{I}'$) when given an interval DNF $\mathcal{I}'$ containing m' intersecting terms as input. We show that $r(m')$ is at most $2\lceil \log(m') \rceil$ which implies that the readability of $\mathcal{I}$ is at most $4\lceil \log(m) \rceil$.

Given an intersecting interval DNF $\mathcal{I}'$ the procedure REDUCE1($\mathcal{I}'$) divides the problem into subproblems $\mathcal{I}_1$ and $\mathcal{I}_2$ respectively. See Figure 6.1. Note that the subproblems $(\mathcal{I}_1 \setminus \{\mathrm{first}(\mathcal{I}_1), \mathrm{last}(\mathcal{I}_1)\}) \mid_{\phi_1=1}$ and $(\mathcal{I}_2 \setminus \{\mathrm{first}(\mathcal{I}_2), \mathrm{last}(\mathcal{I}_2)\}) \mid_{\phi_2=1}$ in the recursive call are again intersecting since $\mathcal{I}_1$ and $\mathcal{I}_2$ are irredundant. For calculating the readability of the formula computed by REDUCE1($\mathcal{I}'$), consider the case when a variable x_i occurs in both subproblems. We show that if x_i does not occur in ϕ_1 (resp. ϕ_2) then it is necessarily the case that it appears in ϕ_2 (resp. ϕ_1) and thus occurring only once in at least one of the subproblems. Note that since $\mathcal{I}'$ is irredundant, the set ϕ_2 forms the interval $[\mathrm{L}(\mathrm{last}(\mathcal{I}_2)), \mathrm{R}(\mathrm{first}(\mathcal{I}_2))]$. Also observe that since x_i occurs in both subproblems and not in ϕ_1, it must lie in the interval $[\mathrm{R}(\mathrm{first}(\mathcal{I}_1)), \mathrm{R}(\mathrm{last}(\mathcal{I}_1))]$. It is easy to see that the latter interval is a subset of ϕ_2 since $\mathrm{R}(\mathrm{last}(\mathcal{I}_1))$ appears before $\mathrm{R}(\mathrm{first}(\mathcal{I}_2))$ in the ordering of variables because of the definition of $\mathcal{I}_1$ and $\mathcal{I}_2$. Also because of the assumption that $\mathcal{I}$ is intersecting, $\mathrm{L}(\mathrm{last}(\mathcal{I}_2))$ appears before $\mathrm{R}(\mathrm{first}(\mathcal{I}_1))$ in the ordering. So the maximum readability of the formula generated by REDUCE1($\mathcal{I}'$) where $\mathcal{I}'$ consists of m' terms satisfies $r(m') \leq 2 + r(\lceil m'/2 \rceil)$. Solving the recurrences yields the stated bound on the readability of $\mathcal{I}'$. $\square$

6.3.2 (p, q)-intersecting DNF

A monotone Boolean DNF is called (p, q)-*intersecting* if every p of its distinct terms intersect in at most q variables. A quadratic DNF for instance, is $(2, 1)$-intersecting, and a k-DNF, i. e., a DNF where the size of each term is bounded by k is $(2, k-1)$-intersecting. In this section, we give a $(p + q - 1)m^{1-\frac{1}{q+1}}$ bound on the readability of (p, q)-intersecting DNF's containing m terms. Theorem 6.2 implies that this bound is almost tight because by considering $q+1$-uniform DNF's containing $m = \Theta(n^{q+1})$ terms we get the following result.

Corollary 6.4. *For a constant q, let $\mathscr{G}_q$ be the class of monotone Boolean DNF on n variables with m terms such that size of every minterm is exactly $q+1$. The readability of almost all $g \in \mathscr{G}_q$ is $\Omega(m^{1-\frac{1}{q+1}} \log^{-1} n)$.*

Algorithm 11 An algorithm to find $(p+q-1)m^{1-\frac{1}{q+1}}$ readable formula for (p,q)-intersecting DNF consists of m terms

Procedure REDUCE2(ϕ, p, q)**:**
Input: A monotone Boolean (p,q)-intersecting DNF ϕ on variables set V
Output: A $(p+q-1)m^{1-\frac{1}{q+1}}$ readable formula ψ equivalent to ϕ

```
1: $\psi := 0,\ m := |\phi|$
2: while $\exists x \in V$ s.t. $\deg_\phi(x) \geq m^{1-\frac{1}{q+1}}$ do
3:     let $\phi_x = \bigvee_{t \in \phi, x \in t} t$
4:     $\phi := \phi \setminus \phi_x$
5:     if $q > 1$ then
6:         $\psi := \psi \vee (x \wedge \text{REDUCE2}(\phi_x |_{x=1}, p, q-1))$
7:     else
8:         $\psi := \psi \vee (x \wedge (\phi_x |_{x=1}))$
9: return $\phi \vee \psi$
```

Let ϕ be a (p,q)-intersecting monotone Boolean DNF on variables $V = \{x_1, \ldots, x_n\}$. Algorithm 11 works by picking a variable x with high degree in ϕ and recursively computing a formula equivalent to the part of ϕ where x occurs. The algorithm stops when every variable has low degree in the remaining expression. More precisely, for a variable $x \in V$, let ϕ_x be the DNF consisting of terms of ϕ which contain x, i.e. $\phi_x = \bigvee_{t \in \phi, x \in t} t$. Note that if ϕ is (p,q)-intersecting then $\phi_x |_{x=1}$ is $(p, q-1)$-intersecting DNF, so the algorithm recurs when $q > 1$ and otherwise it returns the read-$(p-1)$ formula $x \wedge (\phi_x |_{x=1})$. The next Theorem bounds the readability of the formula generated by the algorithm.

Theorem 6.5. *Given a monotone Boolean DNF μ which is (p,q)-intersecting for $p \geq 2, q \geq 1$. The formula $\mu' = REDUCE2(\mu, p, q)$ is $(p+q-1)m^{1-\frac{1}{q+1}}$ readable and it is equivalent to μ.*

Proof. The proof is by induction on q. When $q = 1$, the while loop in Step 2 ensures that every variable in ϕ has degree less then $\sqrt{m}$ after the loop ends. Moreover, a read-$(p-1)$ formula is added to ψ in each iteration of the while loop. Since there are at most $\sqrt{m}$ iterations, the formula $\phi \vee \psi$ in Step 9 has readability at most $\sqrt{m} + (p-1)\sqrt{m}$.

Now assume that the claim is true for $(p, q-1)$ intersecting DNF, where $q \geq 2$. We prove it for (p,q)-intersecting DNF using similar arguments as in the previous paragraph. After the while loop ends, every variable in the remaining ϕ has degree less then $m^{1-\frac{1}{q+1}}$. Let $m_1, \ldots, m_d$ be the number of terms removed from ϕ in each iteration of the while loop, where d is the number of iterations. Note that d can be bounded from above by $m^{\frac{1}{q+1}}$ since each m_i is at least $m^{1-\frac{1}{q+1}}$. Now, denoting the readability of (p,q)-intersecting DNF on m terms by $r_{p,q}(m)$, we have

$$\begin{aligned}
r_{p,q}(m) &\leq m^{1-\frac{1}{q+1}} + \sum_{i=1}^{d} r_{p,q-1}(m_i) && (6.1)\\
&\leq m^{1-\frac{1}{q+1}} + \sum_{i=1}^{d} \left((p+q-2) m_i^{1-\frac{1}{q}} \right) && (6.2)\\
&\leq m^{1-\frac{1}{q+1}} + (p+q-2) d \left(\frac{\sum_{i=1}^{d} m_i}{d} \right)^{1-\frac{1}{q}} && (6.3)\\
&\leq (p+q-1) m^{1-\frac{1}{q+1}}, && (6.4)
\end{aligned}$$

where we apply induction hypothesis to get Equation (6.2) and use Jensen's inequality to get Equation (6.3).

The correctness of the procedure is straightforward since the invariant that $\phi \vee \psi$ is equal to μ holds after completion of every iteration. $\square$

Note that the algorithm produces a depth q formula. In the next section we will see that we can do much better in this regard for a subclass of (p,q)-intersecting DNF, namely the class of DNF where the size of each term is bounded by a constant k.

6.3.3 k-DNF

A monotone Boolean DNF is called k-DNF if every term in it has size at most k. In this section, we give an algorithm to compute $2km^{1-1/k}$ readable formula of depth three and equivalent to the given k-DNF. We need the following definition.

Definition 6.6. *A* sunflower *of size p and a* core *Y is a collection of sets $S_1, \ldots, S_p$ such that $S_i \cap S_j = Y$ for all $i \neq j$ and none of the sets $S_i \setminus Y$ is empty.*

Note that we allow the core Y to be empty, so every pairwise disjoint family of sets constitutes a sunflower.

Lemma 6.7 (Sunflower Lemma [ER60]). *Let $\mathcal{H} \subseteq 2^V$ be a hypergraph with $m = |\mathcal{H}|$ and size of each edge is bounded by k. If $m > k!p^k$ then $\mathcal{H}$ contains a sunflower with $p+1$ petals.*

Since a sunflower has a straightforward read-once representation, the above lemma can be used to give an upper bound on the readability of k-DNF with m terms. The algorithm works by finding a sunflower with certain minimal size, representing it as read-once formula, and then recursing on the remaining edges.

Theorem 6.8. *Let f be a monotone Boolean DNF with m terms such that the size of each term in f is bounded by k then f is $2km^{1-1/k}$-readable. Moreover, a formula of such readability and depth 3 can be found in polynomial time.*

Proof. Any k-DNF with m terms contains a sunflower of size at least $(m/k!)^{1/k}$ which we remove and recurse on the remaining terms. Let $r(m)$ denote the readability of Boolean k-DNF with m terms. Then the readability of f can be bounded by the recurrence $r(m) \leq 1 + r(m - (m/k!)^{1/k})$ with $r(2) = r(1) = 1$. By using the inequality $k! \leq k^k$ and substituting $r(m) \leq 2km^{1-1/k}$ in the above recurrence, it is enough to show that $g(k,m) = 2km^{1-\frac{1}{k}}\left(1 - \left(1 - \frac{m^{\frac{1}{k}-1}}{k}\right)^{1-\frac{1}{k}}\right) \geq 1$. Assume the following claim, which we prove later.

Claim 6.9. *For $k \geq 2$ and $m \geq 1$, the function $g(k,m)$ is monotonically non-increasing in m and monotonically non-decreasing in k.*

Thus the minimum of g is attained when $k = 2$ and m approaches infinity. The minimum value is 1 and hence $r(m) \leq 2km^{1-1/k}$. Finally, we note that the proof of Lemma 6.7 is constructive and a sunflower of desired size can be computed in time polynomial in the number of variables and terms of a DNF. □

Proof of Claim 6.9. Assume for the rest of the proof that $k \geq 2$ and $m \geq 1$. We first observe that the derivative of g with respect to k is positive and so g is non-decreasing in k. Substituting $x = m^{-1+\frac{1}{k}}/k$, we obtain,

$$\begin{aligned}
\frac{\partial g}{\partial k} =& 2\left(1-(1-x)^{1-\frac{1}{k}}\right)m^{1-\frac{1}{k}} + \frac{2\left(1-(1-x)^{1-\frac{1}{k}}\right)\log(m)m^{1-\frac{1}{k}}}{k} \\
&- 2k\left(1-x\right)^{1-\frac{1}{k}}\left(\frac{\left(1-\frac{1}{k}\right)\left(\frac{x}{k}+\frac{x\log m}{k^2}\right)}{1-x} + \frac{\log\left(1-x\right)}{k^2}\right)m^{1-\frac{1}{k}} \\
=& 2m^{1-\frac{1}{k}}\left(\left(1-(1-x)^{1-\frac{1}{k}}\right)\left(1+\frac{\log m}{k}\right)\right. \\
&\left. - k(1-x)^{1-\frac{1}{k}}\left(\frac{x\left(1-\frac{1}{k}\right)\left(1+\frac{\log m}{k}\right)}{k(1-x)} + \frac{\log\left(1-x\right)}{k^2}\right)\right) \\
=& 2m^{1-\frac{1}{k}}\left(1+\frac{\log m}{k}\right) \\
&\left(\left(1-(1-x)^{1-\frac{1}{k}}\right) - \left(\frac{x(1-\frac{1}{k})}{(1-x)^{\frac{1}{k}}} + \frac{\log(1-x)(1-x)^{1-\frac{1}{k}}}{k(1+\frac{\log m}{k})}\right)\right) \\
=& 2m^{1-\frac{1}{k}}\left(1+\frac{\log m}{k}\right)\left(1-\frac{1-2x+\frac{x}{k}}{(1-x)^{\frac{1}{k}}} - \frac{\log(1-x)(1-x)^{1-\frac{1}{k}}}{k(1+\frac{\log m}{k})}\right)
\end{aligned}$$

Only the last factor in the above equation has negative terms. Note that $x = m^{-1+\frac{1}{k}}/k$ is always positive and is at most $1/2$ when $k \geq 2$ and $m \geq 1$. Consequently, the term $(1-2x+\frac{x}{k})/(1-x)^{\frac{1}{k}}$ has value at most 1 and so results in a non-negative quantity when subtracted from 1. Furthermore, the factor $\log(1-x)$ is non-positive and therefore makes the last term non-negative as well.

Similarly, the derivative of g with respect to m is

$$\begin{aligned}\frac{\partial g}{\partial m} =& 2\left(1-\frac{1}{k}\right)\left(k\left(1-\left(1-\frac{m^{\frac{1}{k}-1}}{k}\right)^{1-\frac{1}{k}}\right)m^{-1/k}+\frac{\left(\frac{1}{k}-1\right)\left(1-\frac{m^{\frac{1}{k}-1}}{k}\right)^{-1/k}}{m}\right)\\ =&\frac{2\left(1-\frac{1}{k}\right)}{m\left(1-\frac{m^{\frac{1}{k}-1}}{k}\right)^{\frac{1}{k}}}\left(km^{1-\frac{1}{k}}\left(\left(1-\frac{m^{\frac{1}{k}-1}}{k}\right)^{\frac{1}{k}}+\left(1-\frac{m^{\frac{1}{k}-1}}{k}\right)\right)+\frac{1}{k}-1\right)\end{aligned}$$

which is non-positive since the term $km^{1-\frac{1}{k}}\left(\left(1-\frac{m^{\frac{1}{k}-1}}{k}\right)^{\frac{1}{k}}+\left(1-\frac{m^{\frac{1}{k}-1}}{k}\right)\right)$ is non-decreasing and approaches $\frac{1}{k}-1$ as m goes to infinity. $\square$

6.4 Hardness and Inapproximability

In this section, we show that finding the readability of a given monotone Boolean formula is NP-hard. The reduction we use is gap-introducing and so it also gives hardness of approximating readability unless P = NP. Our reduction is from the well-known NP-complete problem of deciding satisfiability of a given Boolean 3-CNF $\Phi(x_1 \dots x_n) = \bigwedge_{j=1}^{m} \Phi_j$. For all $i \in [n]$ and $j \in [m]$, let us define new variables y_{ij}, y'_{ij}, z'_{ij} for a literal x_i in clause Φ_j and variables z_{ij}, z'_{ij}, y'_{ij} for a literal $\neg x_i$ in clause Φ_j. Let $\phi(y_{11} \dots y_{nm}, z_{11} \dots z_{nm})$ be the monotone CNF we get from $\Phi(x_1 \dots x_n)$ by substituting y_{ij} for x_i in Φ_j and $z_{i'j}$ for $\neg x_{i'}$ in Φ_j such that $\phi(y, z) \equiv \Phi(x)$, for $y_{ij} = x_i$ and $z_{ij} = \neg x_i,\ i \in [n], j \in [m]$. Furthermore, let $I_i = \{j : x_i \in \Phi_j\} \cup \{j : \neg x_i \in \Phi_j\}$, and define

$$\rho(y', z') = \bigwedge_{i=1}^{n}\left(\bigwedge_{j\in I_i} y'_{ij} \vee \bigwedge_{j\in I_i} z'_{ij}\right), \quad \psi(y, z, y', z') = \bigvee_{x_i\in\Phi_j} y_{ij}z'_{ij} \vee \bigvee_{\neg x_i\in\Phi_j} y'_{ij}z_{ij}.$$

Now consider the following Boolean function

$$f(y, z, y', z') = \Big(\phi(y, z) \bigwedge \rho(y', z')\Big) \bigvee \psi(y, z, y', z'). \tag{6.5}$$

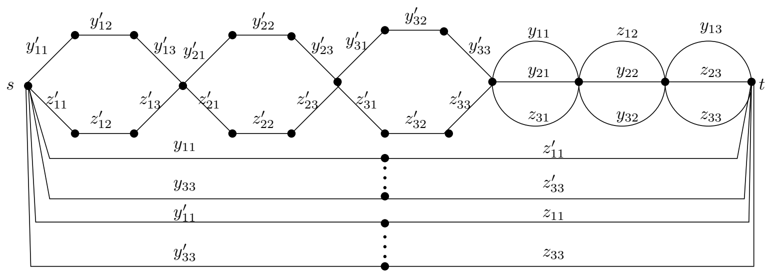

Figure 6.2: Applying the reduction in Equation (6.5) to 3-CNF $\Phi = (x_1 \vee x_2 \vee \neg x_3)(\neg x_1 \vee x_2 \vee x_3)(x_1 \vee \neg x_2 \vee \neg x_3)$. Minimal s-t paths in the figure correspond to minterms of f, whereas minimal s-t cuts are maxterms of f.

Note that the size of f is $15m$, where m is the number of clauses in Φ. The next lemma shows that finding the readability of Boolean formula f defined in Equation (6.5) is equivalent to solving satisfiability for $\Phi(x)$.

Lemma 6.10. *The monotone Boolean function f in Equation* (6.5) *is read-2 if and only if $\Phi(x)$ is satisfiable. It is read-once otherwise.*

Proof. Denote the two disjuncts in f by $f_1 = \phi(y, z) \wedge \rho(y', z')$ and $f_2 = \psi(y, z, y', z')$. We first show that the minterms of f_1 which are not absorbed by minterms of f_2 correspond precisely to the satisfiable assignments of Φ and so $f = \psi$ is clearly a read-once function if Φ is not satisfiable.

Let $\hat{x}$ be a satisfiable assignment of $\Phi(x)$. Since $\hat{x}$ makes at least one literal true in each clause of $\Phi(x)$, the set $t_\phi = \{y_{ij} | \hat{x}_i = 1\} \cup \{z_{ij} | \hat{x}_i = 0\}$ contains a minterm t'_ϕ of $\phi(y, z)$. Similarly, note that the set $t_\rho = \{y'_{ij} | \hat{x}_i = 1\} \cup \{z'_{ij} | \hat{x}_i = 0\}$ defines a minterm of $\rho(y', z')$, and so the set $t = t'_\phi \cup t_\rho$ is a minterm of f_1. It is easy to check that t does not contain any minterm of f_2 since for all $i \in [n]$ and $j \in [m]$, at most one from each pair y_{ij}, z'_{ij} and y'_{ij}, z_{ij} are members of t.

Conversely, any minterm t of f_1 contains one of $y'_{i1}, \ldots, y'_{im}$ or $z'_{i1}, \ldots, z'_{im}$ for all $i \in [n]$ to cover the conjunct ρ. Assume t is not absorbed by any term of f_2. Consequently, t does not contain both $y_{ij}z'_{ij}$ or $y'_{ij}z_{ij}$ for all $i \in [n]$ and $j \in [m]$. Therefore it must contain, from each clause ϕ_j, at least one of the variables y_{ij} or z_{ij} consistent with the primed variable selected from ρ. Hence the assignment $x_i = 1$ if $y_{ij} \in t$ and $x_i = 0$ if

$z_{ij} \in t$ satisfies $\Phi(x)$.

It only remains to prove that f is not a read-once function when $\Phi(x)$ is satisfiable. Assume without loss of generality that the variable x_1 appears in clause Φ_1. Let us define a maxterm c of f by $c = \{y'_{11}, z'_{11}\} \bigcup_{i\in[n],j\in[m]}\{y_{ij}, y'_{ij}\}$ and consider the minterm t of f corresponding to a satisfiable assignment $\hat{x}$ of Φ as defined above. It is easy to see that $|t \cap c| > 1$ since for any literal x_i that appears in clause Φ_j such that $\hat{x}_i = 1$, t would contain both y_{ij} and y'_{ij}. Hence f is not a read-once function because of Theorem 6.1. Note that it is read-2 since we have Equation (6.5) as which gives a read-2 representation. □

Since f in Equation (6.5) is composed of two read-once formulae, Lemma 6.10 implies the hardness of determining if a given monotone formula is disjunction of two read-once formulae .

Corollary 6.11. *It is* NP*-hard to decide whether a given disjunction of two monotone read-once functions is a read-once function.*

A weaker form of Corollary 6.11 can be deduced from the constructions in [GK99] and [BEGM08], which can be used to show the NP-hardness of deciding if a given read-3 monotone formula is actually read-once.

Another interesting problem for which we get a hardness result as a corollary of Lemma 6.10 is the problem of generating all minterms or maxterms of a given monotone Boolean formula. Note that the problem can be solved in polynomial time [GG09] when the input formula is read-once.

Lemma 6.12. *Let $\mathscr{F}$ be the class of monotone Boolean formulae in which each variable appears at most twice. For a formula $f \in \mathscr{F}$, let C_f and D_f denote the sets of the maxterms and the minterms of f, respectively.*

(i) Given a formula $f \in \mathscr{F}$ and a subset C' of C_f, it is coNP*-complete to decide whether $C' = C_f$.*

(ii) Similarly, for a formula $f \in \mathscr{F}$ and a subset D' of D_f, it is coNP*-complete to decide whether $D' = D_f$.*

Proof. Note that since the class $\mathscr{F}$ is closed under duality, both parts of the theorem are equivalent. The hardness of (ii) is implied immediately from Lemma 6.10 by setting $D' = \{t | t \text{ is a term in } \psi\}$. The (possibly) remaining minterms in $D_f \setminus D'$ correspond to satisfiable assignments of Φ. $\square$

In the following, we generalize the reduction introduced in Equation (6.5) and get an inapproximability result for the problem of computing the readability of a given monotone Boolean formula. We use a result of Gál [GÓ2] that gives an explicit monotone Boolean function α on s variables such that the size of the shortest monotone formula representing α is $s^{\Omega(\log s)}$, moreover its irredundant monotone DNF has size $s^{O(\log s)}$. Note that the readability of α is also $s^{\Omega(\log s)}$, since otherwise we could represent α by a formula with smaller than the shortest possible size. We define the following reduction

$$f'(w, y, z, y', z') = \Big(\phi(y, z) \bigwedge \rho(y', z') \bigwedge \alpha(w)\Big) \bigvee \psi(y, z, y', z'),$$

where the size of f' is $15m + s^{O(\log s)}$. Note that if Φ is satisfiable, f' has readability $s^{\Theta(\log s)}$ by applying the same reasoning as in Lemma 6.10. By choosing s and m such that $m = s^{c_1 \log s}$ and $m = c_2 n$ for suitable constants c_1, c_2, we get the following.

Corollary 6.13. *There is no polynomial-time algorithm to approximate the readability of a given monotone Boolean formula f within factor of $O(n)$, unless* P = NP.

6.5 Conclusions

An important unresolved question is whether we can characterize read-k CNF for $k \geq 2$. In contrast, we show in this chapter that for a general monotone Boolean formula, it is NP-hard to decide if it represents a read-once function.

To the best of our knowledge, we were also the first to study the readability of monotone Boolean CNF in terms of the number of terms in it. It would be interesting to know more classes of monotone Boolean CNF's for which small readability representations are efficiently computable.

Chapter 7

Algorithm Engineering

In this chapter we discuss an implementation of a polynomial space algorithm of Elbassioni [Elb08] for the hypergraph transversal problem. The distinguishing feature of our implementation is that it requires polynomial space with the same bound on the running time as the current best. In contrast, all of the previous implementations either have exponential worst-case running time or need super-polynomial space. We found our implementation to be competitive with all but one previous implementation on various datasets.

This chapter is organized as follows. The basic algorithm is described in Section 7.1, while Section 7.2 discusses how this algorithm can be made to run in polynomial space. Next, we present and compare our implementation with some previous implementations in Section 7.3. Finally, we conclude the chapter in Section 7.4.

7.1 The Basic Algorithm

For a hypergraph $\mathcal{H}$ on a set V and a positive number $\epsilon \in (0,1)$, denote by $L = L(\mathcal{H}, \epsilon)$ the subset $\{v \in V : \deg_{\mathcal{H}}(v) > \epsilon|\mathcal{H}|\}$ of "high" degree vertices in $\mathcal{H}$. Given $\epsilon', \epsilon'' \in (0,1)$, let us call an (ϵ', ϵ'')-*balanced set* with respect to $\mathcal{H}$, any set $S \subseteq V$ such that $\epsilon'|\mathcal{H}| \leq |\mathcal{H}_S| \leq \epsilon''|\mathcal{H}|$.

Proposition 7.1. *Let $\epsilon_1, \epsilon_2 \in (0,1)$ be two given numbers such that $\epsilon_1 < \epsilon_2$ and $L = L(\mathcal{H}, \epsilon_1)$ satisfies $|\mathcal{H}_L| \leq (1-\epsilon_2|)|\mathcal{H}|$. Then there exists a $(1-\epsilon_2, 1-(\epsilon_2-\epsilon_1))$-balanced set $L' \supseteq L$.*

The algorithm is based on the following two propositions.

Proposition 7.2. *Let $\mathcal{H}$ be a non-trivial hypergraph on a set V, and $L \subseteq V$ be a given subset of vertices such that $\mathcal{H}_L \neq \emptyset$. Then*

$$\mathcal{H}^d = \bigvee_{v \in L} \left(\left(\mathcal{H}_{V \setminus \{v\}}\right)^d \wedge \{\{v\}\} \right).$$

Proposition 7.3. *Let $\mathcal{H}, \mathcal{F}$ and $\mathcal{G}$ be hypergraphs on a set V such that $\mathcal{H} = \mathcal{F} \vee \mathcal{G}$. Then*

$$\mathcal{H}^d = \bigvee_{Y \in \mathcal{F}^d} \left((\mathcal{G}_{\bar{Y}})^d \wedge \{Y\} \right).$$

Algorithm 12 An algorithm to dualize a given hypergraph

```
Procedure Gen($\mathcal{H}, V, k, \epsilon_1$):
Input: A Sperner hypergraph $\mathcal{H} \subseteq 2^V$ and an integer $k \geq 1$
Output: $\min\{k, |\mathcal{H}^d|\}$ minimal transversals of $\mathcal{H}$
 1: if $|\mathcal{H}| \leq 1$ then
 2:     return $\mathcal{H}^d$
 3: $\epsilon_2 := 2\epsilon_1, L := L(\mathcal{H}, \epsilon_1)$
 4: if $|\mathcal{H}_L| \geq 1$ then
 5:     return $\bigvee_{v \in L} \left( \left(\mathcal{H}_{V \setminus \{v\}}\right)^d \wedge \{\{v\}\} \right)$
 6: $L' := (1-\epsilon_2, 1-(\epsilon_2-\epsilon_1))$-balanced superset of $L$
 7: $\mathcal{Y} := (\mathcal{H}_{L'})^d$
 8: return $\bigvee_{Y \in \mathcal{Y}} \left( (\mathcal{H}_{\bar{Y}})^d \wedge \{Y\} \right)$
```

Given a Sperner hypergraph $\mathcal{H}$ on a set V and an integer $k \geq 1$, we set ϵ_1 to be a suitable constant in $(0, 1/2)$ and fix $\epsilon_2 = 2\epsilon_1$ (In [Elb08], ϵ_1 was set to a function $\epsilon(n,k)$ to optimize the number of recursive calls). To generate k (or all if $k > |\mathcal{H}^d|$) elements of $\mathcal{H}^d$, we call procedure **Gen**$(\mathcal{H}, V, k, \epsilon_1)$ shown as Algorithm 12.

Note that the way the procedure is stated requires that the computation of the transversal hypergraph at each subproblem must be completed before the next subproblem is initiated. This implies that the space required at each recursion-tree node is a function of the size of the output computed at each node. This is in fact unnecessary and can be avoided with a more careful implementation as will be seen in the next section.

7.2 Implementation with Polynomial Space

Consider the recursion tree $\mathbf{T}$. An execution of the algorithm visits the nodes of $\mathbf{T}$ in a certain order, which we call the *execution order*. We classify each node of $\mathbf{T}$ into one of the four types, say 0, 1, 2, or 3, corresponding to which case of the algorithm applies at that node: Type 0 node corresponds to a leaf node, i. e., a subproblem in which the input hypergraph satisfies $|\mathcal{H}| \leq 1$. Type 1 node corresponds to a subproblem of computing $(\mathcal{H}_{V\setminus\{v\}})^d$ for some $L \subseteq V$ and $v \in L$. Type 3 node corresponds to a subproblem of computing $(\mathcal{H}_{L'})^d$ for some $L' \subseteq V$. Finally, type 2 node corresponds to a subproblem of computing $(\mathcal{H}_{\bar{Y}})$ for some $L' \in V$ and $Y \in (\mathcal{H}_{L'}.)$

Nodes of the tree will be generated as needed, and at any time during execution, only a binary subtree $\mathbf{T}'$, called the *active subtree*, is maintained (with possibly some internal nodes of degree only 1) rather than the whole tree $\mathbf{T}$. The active subtree is composed of those nodes of $\mathbf{T}$ that have been visited but whose execution has not yet terminated. Each node in $\mathbf{T}'$ has at most two children with at most one of type 1, and at most one of type 3. Associated with each active node $u \in \mathbf{T}'$ is a *process* $P(u)$, which is created when u is visited for the first time, and is responsible for executing the corresponding subproblem for computing $\mathcal{H}^d$ at u, At any moment of time, only one process, called *current*, is being executed. Execution proceeds normally within the current process P until one of the following two events happens:

(E1) a recursive call is made: in this case a child process of P is created and is made the current process, and the current status of P is saved.

(E2) a **return** statement is reached: in this case P returns a minimal transversal $Y \in \mathcal{H}^d$ (where $\mathcal{H}$ is the input hypergraph of P) to the parent process, or terminates if there is not such element. The parent process becomes then the current process.

When a minimal transversal is returned to the root of the active tree, it gets output. Then a new traversal of the active tree $\mathbf{T}'$ is performed. Any such traversal is performed in *post-order*, that is, for each node we first visit the right child(if there is a child) then the node itself, and then the left child. When a node u of $\mathbf{T}'$ becomes current, execution of the corresponding process $P(u)$ proceeds, from the last point at which it was stopped. Thus, when $P(u)$ returns a minimal transversal, it returns the next element of $\mathcal{H}^d$ that has not been returned yet, if there is such an element. When a process terminates, it gets deleted from the active tree.

Since the algorithm outputs the minimal transversals one by one, we have to modify the disjunction computation to return one transversal at a time. This can be done as follows.

- Type 1 node: suppose that u corresponds to a subproblem of computing $(\mathcal{H}_{V\setminus\{v\}})^d$, for some $L \subseteq V$ and $v \in L$ (step 5). Suppose further that u is ready to return the minimal transversal $X \in (\mathcal{H}_{V\setminus\{v\}})^d$. Before $P(u)$ returns X, it must check whether $X \cup \{v\}$ really belongs to the disjunction $\bigvee_{v\in L}\left((\mathcal{H}_{V\setminus\{v\}})^d \wedge \{\{v\}\}\right)$. A necessary and sufficient condition for this is that $X \cup \{v\}$ is a minimal transversal of $\mathcal{H}$ (which is the input hypergraph at parent node). To avoid producing a minimal transversal more than once, we also have to check that $X \cup \{v\}$ does not contain v' for any $v' \in L$, whose corresponding subproblem has been considered before at the parent node.

- Type 3 node: suppose that u corresponds to a subproblem of computing $(\mathcal{H}_{\bar{Y}})^d$, for some $L' \subseteq V$ and some $Y \in (\mathcal{H}_{L'})^d$ (step 7). When $P(u)$ is ready to return X, we must check first that $X \cup Y$ is a minimal transversal to $\mathcal{H}$ (which is the input hypergraph at the parent node). We also have to account for the fact that different pairs (X, Y) can produce the same minimal transversal. To avoid such repetition, we assume an *arbitrary ordering* on the vertices of V. When a minimal transversal

X is found we return $X \cup Y$ to the parent node only if Y is the *lexicographically last* subset of $X \cup Y$ which is in $(H_{L'})^d$ (such a subset can be found in $O(|X \cup Y||H_{L'}|)$ time).

In the implementation given as Algorithm 13, we set ϵ_1 and ϵ_2 as before. Given the current node u of the active tree, we assume that the two calls $Y :=$ **Gen-Next**$(\ldots)$, $X :=$ **Gen-Next**$(\ldots)$ in step 13 and 14 correspond respectively to the left and right children of u (so that the post-order traversal will find all possible such Y's before finding the next X). To simplify the presentation, we assume implicitly that the procedure terminates once k minimal transversal are returned.

Algorithm 13 A polynomial-space algorithm to dualize a given hypergraph.

```
Procedure Gen-Next($\mathcal{H}, V, k, \epsilon_1$):
Input: A Sperner hypergraph $\mathcal{H} \subseteq 2^V$ and an integer $k \geq 1$
Output: $\min\{k, |\mathcal{H}^d|\}$ minimal transversals of $\mathcal{H}$
 1: if $|\mathcal{H}| = 0$ then
 2:     return $\{\emptyset\}$
 3: else if $\mathcal{H} = \{H\}$ then
 4:     return "next" element of $\{\{i\} : i \in H\}$
 5: $\epsilon_2 := 2\epsilon_1$, $L := L(\mathcal{H}, \epsilon_1) \overset{\text{def}}{=} \{v_1, \ldots, v_l\}$
 6: if $|\mathcal{H}_L| \geq 1$ then
 7:     for $i = 1 \ldots, l$ do
 8:         $W :=$ Gen-Next$(\mathcal{H}_{V \setminus \{v_i\}}, V \setminus \{v_i\}, k)$; $T := W \cup \{v_i\}$
 9:         if $T \in \mathcal{H}^d$ and $T \not\ni v_j, \forall j \in [i-1]$ then
10:             return T
11: else
12:     $L' := (1 - \epsilon_2, 1 - (\epsilon_2 - \epsilon_1))$-balanced superset of $L$
13:     $Y :=$ Gen-Next$(\mathcal{H}_{L'}, L', k)$
14:     $X :=$ Gen-Next$(\mathcal{H}_{\bar{Y}}, \bar{Y}, k)$
15:     Compute the lexicographically last set $Z \subseteq X \cup Y$ s.t. $Z \in (\mathcal{H}_{L'})^d$
16:     if $Z = Y$ and $X \cup Y \in \mathcal{H}^d$ then
17:         return $X \cup Y$
```

For correctness and analysis of the running time, we refer to the original publication [Elb08].

7.3 Results

We implemented the algorithm in C++. The STL bitset is chosen to represent subsets, which results in convenient and fast subset operations using C++ bitwise operators. In this section we call our implementation PS. We also added a heuristic in our implementation to merge vertices when they are indistinguishable to each other with respect to hyperedges, that is, we merge vertices x_1 and x_2 into a single vertex when there is no hyperedge H such that $x_1 \in H$ and $x_2 \notin H$. The variant of our implementation with the above heuristic is called $\mathrm{PS_h}$ in this section. We compare our implementation with the following three published implementations.

- **KBEG**: An implementation of quasi-polynomial algorithm of Fredman and Khachyian [FK96] in C/C++. It is presented in [KBEG06]. We have access to the source code from the authors, which we compiled by using the same options as our program.

- **BMR**: A C/C++ implementation based on Berge multiplication of Section 2.4.1 presented in [BMR03]. We got the source code from the authors and compiled it by using the same options as our program

- **KS**: Another implementation based on Berge multiplication and presented in [KS99]. We use the publicly available linux binary for our experiments.

The experiments were run on a 2.4Ghz AMD Opteron equipped with 8GB RAM and running Debian Etch as an operating system. All programs except KS were compiled with g++ 4.1 with optimization level 2. For program KS, we received the compiled binary from its authors. We use the following datasets from [KBEG06] to compare different implementations.

- *Random* ($\alpha(n, m, [l, r])$): The random hypergraph on n vertices and m edges. The edges are generated by repeating the following random experiment. We first pick the size of an edge k uniformly at random from $[l, r]$. Next a random sample of size k is picked from the n vertices. The above experiment is repeated until m irredundant hyperedges are generated.

- *Matching* (M(n)): The matching graph on vertex set $\{1, \ldots, n\}$ containing $n/2$ hyperedges of the form $\{i, i+1\}$ for $i = 1, 3, \ldots, n-1$, where n is even. Its dual hypergraph is clearly exponential in the size of primal.

- *Matching Dual* (DM(n)): The dual hypergraph of M(n) for even n.

- *Threshold graph* (TH(n)): This is a graph on n vertices numbered from 1 to n (where n is even), with edge set $\{\{i, j\} \ : \ 1 \le i < j \le n, \ j \text{ is even }\}$, i. e., for $j = 2, 4, \ldots, n$, there is an edge between i and j for all $i < j$. We are interested in these graphs because they have both small number of edges (namely, $n^2/4$) and small number of minimal transversals (namely, $n/2 + 1$).

- *Self-dualized threshold graph* (SDTH(n)): This is a self-dual hypergraph on n vertices obtained from the threshold graph TH$(n-2)$ and its dual $(\text{TH}(n-2))^d$ as follows:

$$\{\{n-1, n\}\} \bigcup \{\{n-1\} \cup H : H \in \ \text{TH}(n-2)\} \bigcup \left\{\{n\} \cup H : H \in (\text{TH}(n-2))^d\right\}.$$

  This gives a family of hypergraphs with polynomially bounded input and output sizes $|\text{SDTH}(n)| = |(\text{SDTH}(n))^d| = (n-2)^2/4 + n/2 + 1$.

- *Self-dualized Fano-plane product* (SDFP(n)): This is constructed by starting with the hypergraph $\mathcal{H}_0 = \{\{1,2,3\}, \{1,5,6\}, \{1,7,4\}, \{2,4,5\}, \{2,6,7\}, \{3,4,6\}, \{3,5,7\}\}$ (which represents the set of lines in a Fano plane and is self-dual), taking $k = (n-2)/7$ disjoint copies $\mathcal{H}_1, \ldots, \mathcal{H}_k$ of $\mathcal{H}_0$, and letting $\mathcal{H} = \mathcal{H}_1 \cup \ldots \cup \mathcal{H}_k$. The dual hypergraph $\mathcal{H}^d$ is just the hypergraph of all 7^k unions obtained by taking one hyperedge from each of the hypergraphs $\mathcal{H}_1, \ldots, \mathcal{H}_k$. Finally, we define the hypergraph SDFP(n) to be the hypergraph of $1 + 7k + 7^k$ hyperedges on n vertices, obtained by self-dualizing $\mathcal{H}$, as we did for threshold graphs.

$\lvert V\rvert$	$\lvert\mathcal{H}\rvert$	$\lvert\mathcal{H}^d\rvert$	KBEG	BMR	KS	PS	PS$_\text{h}$
20	10	2^{10}	0.21	0.04	0.01	0.10	0.37
24	12	2^{12}	1.42	0.07	0.02	0.08	0.15
28	14	2^{14}	2.73	0.17	0.03	0.31	0.15
30	15	2^{14}	6.62	0.31	0.07	0.66	0.38
32	16	2^{16}	12.58	0.60	0.12	1.35	0.58
34	17	2^{17}	21.80	1.21	0.22	2.79	1.26
36	18	2^{18}	89.23	2.37	0.45	6.14	2.60
38	19	2^{19}	$*$	4.77	0.99	13.13	5.46
40	20	2^{20}	$*$	9.96	1.99	26.33	11.31

Table 7.1: Performance of different implementations on matching graph dataset M(n).

$\lvert V\rvert$	$\lvert\mathcal{H}\rvert$	$\lvert\mathcal{H}^d\rvert$	KBEG	BMR	KS	PS	PS$_\text{h}$
20	2^{10}	10	2.00	0.38	0.03	52.16	36.93
24	2^{12}	12	5.02	3.34	0.15	7,970.51	5,045.60
28	2^{14}	14	14.07	46.26	1.20	$-$	$-$
30	2^{15}	15	45.47	184.94	3.69	$-$	$-$
32	2^{16}	16	146.50	758.22	12.46	$-$	$-$
34	2^{17}	17	$*$	2,102.95	43.72	$-$	$-$
36	2^{18}	18	$*$	11,043.73	150.49	$-$	$-$

Table 7.2: Performance of different implementations on matching dual dataset DM(n).

In the following experiments, we set $\epsilon_1 = 0.2$ in our implementation. Table 7.1 presents the running time taken by different implementations on dataset M(n) in seconds. Our implementation was considerably faster than KBEG, while the variant PS$_\text{h}$ was comparable to BMR on this dataset. The implementation KS was fastest on all instances of M(n). On the matching dual dataset DM(n) (Table 7.2) our implementation was beaten by all other implementations. In fact, it failed to terminate in less the two hours on larger instances of DM(n). In Table 7.1 and the following tables, we denote by the symbols $*$ or $-$ that the program crashed or did not terminate in reasonable amount of time, respectively.

The performance of different implementations on threshold dataset T(n) is given in Table 7.3. Out implementation was faster than BMR. While other implementations have performed better than both PS and PS$_\text{h}$.

$\lvert V\rvert$	$\lvert\mathcal{H}\rvert$	$\lvert\mathcal{H}^d\rvert$	KBEG	BMR	KS	PS	PS$_\text{h}$
40	400	21	0.87	0.10	0.04	0.07	0.06
60	900	31	1.49	0.51	0.13	0.31	0.35
80	1500	41	2.46	2.12	0.46	0.98	0.95
100	2500	51	3.62	6.51	1.25	2.46	2.72
120	3600	61	5.61	15.66	2.93	5.82	5.56
140	4900	71	8.47	33.77	6.23	12.98	11.79
160	6400	81	15.01	66.54	12.01	26.52	22.98
180	8100	91	23.98	121.28	21.81	48.72	42.22
200	10000	101	37.68	213.15	38.02	83.25	70.95

Table 7.3: Performance of different implementations on threshold graphs dataset (TH(n)).

$\lvert V\rvert$	$\lvert\mathcal{H}\rvert = \lvert\mathcal{H}^d\rvert$	KBEG	BMR	KS	PS	PS$_\text{h}$
42	422	1.18	0.11	0.06	0.12	0.17
62	932	2.53	0.53	0.23	0.66	0.69
82	1642	6.79	2.15	0.78	2.20	2.06
102	2552	16.02	6.45	2.20	6.04	5.49
122	3662	50.04	15.72	4.84	14.04	12.50
142	4972	74.20	33.68	10.08	30.08	26.24
162	6482	144.76	66.69	19.97	58.06	50.17
182	8192	572.40	122.25	37.11	105.77	90.61
202	10102	451.59	213.32	64.48	183.40	154.51

Table 7.4: Performance of different implementations on self dualized threshold graph dataset (SDTH(n)).

On self-dualized threshold dataset SDTH(n) (Table 7.4), our implementation have beaten KBEG and BMR, while KS has toped again on this dataset.

Table 7.5 presents the running times of different implementations on self-dualized Fano plane product dataset (SDFP(n)). Out implementation did perform better than BEGK on small instances, but failed to terminate in less than 24 hours on large instances.

Finally, the results of our experiments on random hypergraphs $\alpha(n, m, [l, r])$ are presented in Table 7.6. This particular set of experiments were ran on Lenovo T400 laptop powered with Intel P8600 clocked at 2.40GHz, containing 2GB of RAM and running Ubuntu Lucid Lynx as an operating system. Also in these experiments we were not able to setup the KBEG implementation, and so the reults for KBEG are not reported.

$\lvert V\rvert$	$\lvert\mathcal{H}\rvert = \lvert\mathcal{H}^d\rvert$	KBEG	BMR	KS	PS	PS$_\text{h}$
16	64	2.11	0.02	0.02	0.09	0.08
23	365	68.47	0.22	0.14	172.60	30.78
30	2430	1,735.22	9.37	6.72	–	56,649.56
37	16843	–	457.45	510.18	–	–

Table 7.5: Performance of different implementations on self dualized Fano plane product dataset (SDFP(n)).

n	m	$[l,r]$	$\lvert\mathcal{H}^d\rvert$	BMR	KS	PS$_\text{h}$
50	100	$[20,45]$	283131	3.25	7.93	117.81
50	200	$[25,45]$	390835	4.64	17.12	180.07
50	300	$[30,45]$	257268	2.59	14.69	64.57
50	400	$[35,45]$	139471	1.27	12.57	24.23

Table 7.6: Performance of different implementations on random hypergraphs $\alpha(n, m, [l, r])$.

For each entry in the table, we report the average over 10 randomly generated hypergraphs with the given parameters. Our implementation was trailed by KS and BMR implementations in all cases.

7.4 Conclusions

This chapter presents an implementation of quasi-polynomial time algorithm of Elbassioni [Elb08] that runs in polynomial space. While the algorithm runs provably in quasi-polynomial time in combined input and output size, it fails to beat another implementation which is based on sub-optimal Berge multiplication.

To the best of our knowledge, no implementation with provable quasi-polynomial running time has yet managed to outperform simple heuristic based implementations in general. This raises the natural question about the suitability of quasi-polynomial algorithms in practice. The instance classes which are hard for multiplication need also be studied. Currently, the only known class which is hard for multiplication grows double exponentially [Tak07], making it intractable to solve for comparison purposes.

List of Algorithms

List of Figures

List of Tables

Bibliography

[AB92] M. Anthony and N. Biggs. *Computational learning theory: an introduction*. Cambridge University Press, New York, NY, USA, 1992.

[AB98] M. Akra and L. Bazzi. On the solution of linear recurrence equations. *Comput. Optim. Appl.*, 10(2):195–210, 1998.

[AHK93] D. Angluin, L. Hellerstein, and M. Karpinski. Learning read-once formulas with queries. *J. ACM*, 40(1):185–210, 1993.

[AIS93] R. Agrawal, T. Imieliński, and A. Swami. Mining association rules between sets of items in large databases. In *SIGMOD '93: Proceedings of the 1993 ACM SIGMOD international conference on Management of data*, pages 207–216, New York, NY, USA, 1993. ACM.

[AJHL89] M. Albertson, R. Jamison, S. Hedetniemi, and S. Locke. The subchromatic number of a graph. *Discrete Mathematics*, 74(1-2):33–49, 1989.

[AMS$^+$96] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. I. Verkamo. Fast discovery of association rules. pages 307–328, 1996.

[AS94] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *Proc. VLDB'94*, pages 487–499, 1994.

[BEG$^+$02] E. Boros, K. Elbassioni, V. Gurvich, L. Khachiyan, and K. Makino. Dual-bounded generating problems: All minimal integer solutions for a monotone system of linear inequalities. *SIAM J. Computing*, 31(5):1624–1643, 2002.

[BEGK04] E. Boros, K. Elbassioni, V. Gurvich, and L. Khachiyan. Generating maximal independent sets for hypergraphs with bounded edge-intersections. In *LATIN '04*, pages 488–498, 2004.

[BEGM08] E. Boros, K. Elbassioni, V. Gurvich, and K. Makino. Generating vertices of polyhedra and related monotone generation problems. In David Avis, David Bremner, and Antoine Deza, editors, *CRM Proceedings & Lecture Notes, Centre de Recherches Mathématiques at the Université de Montréal, special issue on Polyhedral Computation*, volume 48, pages 15–39. American Mathematical Society, 2008.

[BEM08] E. Boros, K. M. Elbassioni, and K. Makino. On berge multiplication for monotone boolean dualization. In *ICALP (1)*, pages 48–59, 2008.

[BEM10] E. Boros, K. M. Elbassioni, and K. Makino. Left-to-right multiplication for monotone boolean dualization. *SIAM J. Comput.*, 39(7):3424–3439, 2010.

[Ber89] C. Berge. *Hypergraphs*. Elsevier-North Holand, Amsterdam, 1989.

[BGH98] E. Boros, V. Gurvich, and P.L. Hammer. Dual subimplicants of positive boolean functions. *Optim. Methods Softw.*, 10:147–156, 1998.

[BGKM02] E. Boros, V. Gurvich, L. Khachiyan, and K. Makino. On the complexity of generating maximal frequent and minimal infrequent sets. In *STACS '02*, pages 133–141, 2002.

[BI95] J. C. Bioch and T. Ibaraki. Complexity of identification and dualization of positive boolean functions. *Information and Computation*, 123(1):50–63, 1995.

[BM09] E. Boros and K. Makino. A fast and simple parallel algorithm for the monotone duality problem. In *ICALP '09*, 2009.

[BMR03] J. Bailey, T. Manoukian, and K. Ramamohanarao. A fast algorithm for computing hypergraph transversals and its application in mining emerging patterns. In *Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM 2003)*, pages 485–488, 2003.

[Cha93] B. Chazelle. Cutting hyperplanes for divide-and-conquer. *Discrete & Computational Geometry*, 9:145–158, 1993.

[Col87] C. J. Colbourn. *The Combinatorics of Network Reliability*. Oxford University Press, NY, USA, 1987.

[dBvKOS97] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry, Algorithms and Applications*. Springer Verlag, Amsterdam, 1997.

[DF99] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer, 1999.

[DMP99] C. Domingo, N. Mishra, and L. Pitt. Efficient read-restricted monotone CNF/DNF dualization by learning with membership queries. *Machine Learning*, 37(1):89–110, 1999.

[EG95] T. Eiter and G. Gottlob. Identifying the minimal transversals of a hypergraph and related problems. *SIAM J. Computing*, 24(6):1278–1304, 1995.

[EGM02] T. Eiter, G. Gottlob, and K. Makino. New results on monotone dualization and generating hypergraph transversals. In *STOC '02*, pages 14–22, 2002.

[EGM03] T. Eiter, G. Gottlob, and K. Makino. New results on monotone dualization and generating hypergraph transversals. *SIAM J. Computing*, 32(2):514–537, 2003.

[EHR08] K Elbassioni, M. Hagen, and I. Rauf. Some fixed-parameter tractable classes of hypergraph duality and related problems. In *IWPEC*, pages 91–102, 2008.

[Eit94] T. Eiter. Exact transversal hypergraphs and application to Boolean μ-functions. *Journal of Symbolic Computation*, 17(3):215–225, 1994.

[Elb06] K. Elbassioni. On the complexity of the multiplication method for monotone CNF/DNF dualization. In *ESA '06*, pages 340–351, 2006.

[Elb08] K. M. Elbassioni. On the complexity of monotone dualization and generating minimal hypergraph transversals. *Discrete Applied Mathematics*, 156(11):2109–2123, 2008.

[EMG08] T. Eiter, K. Makino, and G. Gottlob. Computational aspects of monotone dualization: A brief survey. *Discrete Applied Mathematics*, 156(11):2035–2049, 2008.

[EMR09] K. Elbassioni, K. Makino, and I. Rauf. Output-sensitive algorithms for enumerating minimal transversals for some geometric hypergraphs. In *ESA*, 2009.

[ER60] P. Erdös and R. Rado. Intersection theorems for systems of sets. *J. London Math. Soc.*, 35:85–90, 1960.

[FK96] M. L. Fredman and L. Khachiyan. On the complexity of dualization of monotone disjunctive normal forms. *Journal of Algorithms*, 21:618–628, 1996.

[FS91] P. Fischer and H. U. Simon. Separation problems and circular arc systems. In *WG '90: Proceedings of the 16rd International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 251–259, London, UK, 1991. Springer-Verlag.

[Ǵ02] A. Gál. A characterization of span program size and improved lower bounds for monotone span programs. *Comput. Complex.*, 10(4):277–296, 2002.

[GG09] M. C. Golumbic and V. Gurvich. Read-once functions. In Yves Crama and Peter Ladislaw Hammer, editors, *Boolean Functions: Theory, Algorithms and Applications*. Cambridge University Press, 2009. (in press).

[GIK02] D. R. Gaur, T. Ibaraki, and R. Krishnamurti. Constant ratio approximation algorithms for the rectangle stabbing problem and the rectilinear partitioning problem. *J. Algorithms*, 43(1):138–152, 2002.

[GK99] V. Gurvich and L. Khachiyan. On generating the irredundant conjunctive and disjunctive normal forms of monotone Boolean functions. *Discrete Applied Mathematics*, 96-97(1):363–373, 1999.

[GK04] D. R. Gaur and R. Krishnamurti. Average case self-duality of monotone boolean functions. In *Canadian AI '04*, pages 322–338, 2004.

[GKM$^+$03] D. Gunopulos, R. Khardon, H. Mannila, S. Saluja, H. Toivonen, and R. S. Sharm. Discovering all most specific sentences. *ACM Trans. Database Syst.*, 28(2):140–174, 2003.

[GMB85] H. Garcia-Molina and D. Barbara. How to assign votes in a distributed system. *Journal of ACM*, 32(4):841–860, 1985.

[GMKT97] D. Gunopulos, H. Mannila, R. Khardon, and H. Toivonen. Data mining, hypergraph transversals, and machine learning (extended abstract). In *PODS '97*, pages 209–216, 1997.

[GMR06] M. C. Golumbic, A. Mintz, and U. Rotics. Factoring and recognition of read-once functions using cographs and normality and the readability of functions associated with partial k-trees. *Discrete Applied Mathematics*, 154(10):1465–1477, 2006.

[Got04] G. Gottlob. Hypergraph transversals. In *FoIKS '04: Proc. of the 3rd International Symposium on Foundations of Information and Knowledge Systems*, pages 1–5, 2004.

[Gur77] V. Gurvich. On repetition-free boolean functions. *Uspekhi Mat. Nauk. (Russian Math. Surveys)*, 32:183–184, 1977. (in Russian).

[Gur91] V. Gurvich. Criteria for repetition-freeness of functions in the algebra of logic. *Soviet Math. Dokl.*, 43(3):721–726, 1991.

[Hag07] M. Hagen. On the fixed-parameter tractability of the equivalence test of monotone normal forms. *Inf. Process. Lett.*, 103(4):163–167, 2007.

[Hag08] M. Hagen. *Algorithmic and Computational Complexity Issues of MONET*. PhD thesis, Friedrich Schiller University of Jena, 2008.

[HNW08] F. Hüffner, R. Niedermeier, and S. Wernicke. Techniques for practical fixed-parameter algorithms. *Comput. J.*, 51(1):7–25, 2008.

[IK93] T. Ibaraki and T. Kameda. A theory of coteries: Mutual exclusion in distributed systems. *IEEE Trans. on Parallel and Distributed Systems*, 4(7):779–794, 1993.

[JPY88] D. S. Johnson, C. H. Papadimitriou, and M. Yannakakis. On generating all maximal independent sets. *Information Processing Letters*, 27(3):119–123, 1988.

[Kal97] M. J. Kaltz. 3-D vertical ray shooting and 2-D point enclosure, range searching, and arc shooting amidst convex fat objects. *Comput. Geom. Theory Appl.*, 8(6):299–316, 1997.

[KBE$^+$05] L. Khachiyan, E. Boros, K. Elbassioni, V. Gurvich, and K. Makino. On the complexity of some enumeration problems for matroids. *SIAM Journal on Discrete Mathematics*, 19(4):966–984, 2005.

[KBEG06] L. Khachiyan, E. Boros, K. M. Elbassioni, and V. Gurvich. An efficient implementation of a quasi-polynomial algorithm for generating hypergraph transversals and its application in joint generation. *Discrete Applied Mathematics*, 154(16):2350–2372, 2006.

[KBEG07a] L. Khachiyan, E. Boros, K. Elbassioni, and V. Gurvich. A global parallel algorithm for the hypergraph transversal problem. *Information Processing Letters*, 101(4):148–155, 2007.

[KBEG07b] L. Khachiyan, E. Boros, K. M. Elbassioni, and V. Gurvich. On the dualization of hypergraphs with bounded edge-intersections and other related classes of hypergraphs. *Theor. Comput. Sci.*, 382(2):139–150, 2007.

[KBGE07] L. Khachiyan, E. Boros, V. Gurvich, and K. Elbassioni. Computing many maximal independent sets for hypergraphs in parallel. *Parallel Processing Letters*, 17(2):141–152, 2007.

[Kha00] L. Khachiyan. Transversal hypergraphs and families of polyhedral cones. In *Advances in Convex Analysis and Global Optimization, honoring the memory of K. Carathéodory*, pages 105–118. Kluwer, 2000.

[KPS93] D. J. Kavvadias, C. H. Papadimitriou, and M. Sideri. On Horn envelopes and hypergraph transversals. In *ISAAC '93*, pages 399–405, 1993.

[KS99] D. Kavvadias and E. C. Stavropoulos. Evaluation of an algorithm for the transversal hypergraph problem. In *WAE '99: Proceedings of the 3rd International Workshop on Algorithm Engineering*, pages 72–84, London, UK, 1999. Springer-Verlag.

[KS03] D. J. Kavvadias and E. C. Stavropoulos. Checking monotone boolean duality with limited nondeterminism. Technical Report CTI TR2003.07.02, Computer Technology Institute, Patras, Greece, 2003.

[KS06] S. Kovaleva and F. C. R. Spieksma. Approximation algorithms for rectangle stabbing and interval stabbing problems. *SIAM Journal on Discrete Mathematics*, 20(3):748–768, 2006.

[KUW88] R. M. Karp, E. Upfal, and A. Wigderson. The complexity of parallel search. *Journal of Computer and System Sciences*, 36(2):225–253, 1988.

[Lau08] S. Laue. Geometric set cover and hitting sets for polytopes in $\mathbb{R}^3$. In *STACS*, pages 479–490, 2008.

[LLK80] E. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan. Generating all maximal independent sets: NP-hardness and polynomial-time algorithms. *SIAM J. Computing*, 9:558–565, 1980.

[Lov92] L. Lovász. Combinatorial optimization: some problems and trends. DIMACS Technical Report 92-53, Rutgers University, 1992.

[Mak03] K. Makino. Efficient dualization of $O(\log n)$-term monotone disjunctive normal forms. *Discrete Applied Mathematics*, 126(2-3):305–312, 2003.

[Mat92] J. Matousek. Efficient partition trees. *Discrete & Computational Geometry*, 8:315–334, 1992.

[MI97] K. Makino and T. Ibaraki. The maximum latency and identification of positive boolean functions. *SIAM J. Computing*, 26(5):1363–1383, 1997.

[MPV05] Y. Manolopoulos, A. Papadopoulos, and M. Vassilakopoulos, editors. *Spatial Databases: Technologies, Techniques and Trends*. Idea Group, 2005.

[MR86] H. Mannila and K. J. Räihä. Design by example: An application of armstrong relations. *Journal of Computer and System Sciences*, 33(2):126–141, 1986.

[Nie06] R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006.

[Pap97] C. Papadimitriou. NP-completeness: A retrospective. In *ICALP '97*, pages 2–6, 1997.

[Ram90] K. G. Ramamurthy. *Coherent Structures and Simple Games*. Kluwer, 1990.

[RT75] R. C. Read and R. E. Tarjan. Bounds on backtrack algorithms for listing cycles, paths, and spanning trees. *Networks*, 5:237–252, 1975.

[Tak07] K. Takata. A worst-case analysis of the sequential method to list the minimal hitting sets of a hypergraph. *SIAM J. Discrete Math.*, 21(4):936–946, 2007.

[Tam00] H. Tamaki. Space-efficient enumeration of minimal transversals of a hypergraph. Technical Report IPSJ-AL 75, 2000.

[Weg87] I. Wegener. *The complexity of Boolean functions*. John Wiley & Sons, Inc., New York, NY, USA, 1987.