DIMACS Technical Report 2006-01

On the Complexity of Monotone Boolean Duality Testing[1]

by

Khaled M. Elbassioni[,2]

[1]This work is dedicated to the memory of Leonid Khachiyan

[2]Max-Planck-Institut für Informatik, Saarbrücken, Germany; (elbassio@mpi-sb.mpg.de)

ABSTRACT

We show that the duality of a pair of monotone Boolean functions in disjunctive normal forms can be tested in polylogarithmic time using a quasi-polynomial number of processors. Our decomposition technique yields stronger bounds on the complexity of the problem than those currently known and also allows for generating all minimal transversals of a given hypergraph using only polynomial space.

1 Introduction

Let $f : \{0,1\}^n \mapsto \{0,1\}$ be a *monotone* Boolean function, defined by its *irredundant disjunctive normal form* (DNF)

$$f(x) = \bigvee_{F \in \mathcal{F}} \bigwedge_{i \in F} x_i,$$

where $\mathcal{F} \subseteq 2^V$ is a *Sperner* hypergraph (i.e. no hyperedge in $\mathcal{F}$ contains another), defined on the set of variable indices $V \stackrel{\text{def}}{=} \{1,\ldots,n\}$, and represents the family of *prime implicants* of f. Such a DNF representation exists and is uniquely defined for any monotone Boolean function. The *dual* of $f = f(x_1,\ldots,x_n)$, denoted by f^d, is defined as $f^d(x) = \bar{f}(\bar{x}_1,\ldots,\bar{x}_n)$, or equivalently by the *conjunctive* normal form (CNF) $f^d(x) = \bigwedge_{F \in \mathcal{F}} \bigvee_{i \in F} x_i$. The *monotone Boolean dualization* problem is to compute, for a given monotone Boolean function f, the prime implicants of the dual function f^d. From these definitions, it follows that if $\mathcal{F}$ is the hypergraph representing the prime implicants of f, then

$$f^d(x) = \bigvee_{F \in \mathcal{F}^d} \bigwedge_{i \in F} x_i,$$

where $\mathcal{F}^d \subseteq 2^V$ is the *dual (or transversal)* hypergraph, i.e. the hypergraph of all *minimal transversals (or minimal hitting sets)* of $\mathcal{F}$. Clearly, the number K of prime implicants of f^d can be exponential in the number n of variables and the number m of prime implicants of f, and hence one can only hope for an algorithm whose efficiency is measured in terms of these parameters n, m and K. In particular, finding f^d can be reduced to $K+1$ calls to the following decision problem:

Duality-testing(f,g)**:** *Given two monotone Boolean functions f, g by their irredundant disjunctive normal forms $\mathcal{F}, \mathcal{G} \subseteq 2^V$, check if $f = g^d$.*

Determining the exact complexity of the duality testing problem is an outstanding open question, which has been referenced in a number of complexity theory retrospectives, e.g. [41, 45], and has been recently the subject of growing research, see e.g. [4, 15, 18, 19, 22, 26, 50]. In a landmark paper, Fredman and Khachiyan [22] gave an algorithm for solving this problem whose running time is $O(nN) + N^{o(\log N)}$, where $N = |\mathcal{F}| + |\mathcal{G}|$, thus providing strong evidence that this decision problem is unlikely to be NP-hard. Eiter, Gottlob and Makino [20], and independently, Kavvadias and Stavropoulos [35] used this result to show that the duality of a monotone pair can be disproved in polynomial time with $O(\log^2 N / \log\log N)$ suitably guessed bits, i.e. with logarithmically limited nondeterminism, putting thus the problem in the class co-β_2P (see e.g. [36] for the definition of this class).

Our main result, stated more precisely in Section 1.1 below and proved in Section 2, places Duality-testing further into the class of problems solvable (in the PRAM model) in *polylogarithmic time* using a *quasi-polynomial* number of processors. This is obtained through a new set of decomposition rules that allow us to obtain a recursion tree with polylogarithmic depth, a feature that is not known to exist in previous algorithms. As a byproduct of this decomposition technique, we also obtain stronger bounds on the complexity of duality testing, which can make use of the possible *assymtery* in the sizes of $\mathcal{F}$ and $\mathcal{G}$: if $m = |\mathcal{F}|$, $K = |\mathcal{G}|$, and $N = m + K$, then we can test duality in time $O(nN) + m^{o(\log k)}$ if $k >> n$ and in time $O(nN) + k^{o(\log m)}$ if $m >> n$, see Theorem 1 below for the precise bounds. A further implication of this result is that, if π is a monotone property defined over the vertices of the n-dimensional hypercube and given by an *oracle*, then the characterizing vectors of the property, i.e. the families of minimal vectors satisfying and maximal vectors not satisfying π, can be listed in randomized $O(\sqrt{n})$ *oracle parallel time* per vector using a quasi-polynomial number of processors; see Section 1.2 for more details.

Since the dualization of a monotone Boolean function f is equivalent to computing the minimal transversals (or equivalently the maximal independent sets) of the hypergraph representing the DNF of f, we may also consider the following incremental generation problem:

GEN-TRANS$(\mathcal{H}, k)$**:** *Given a hypergraph* $\mathcal{H}$*, and an integer* $k \geq 1$*, generate* k *(or all if* $|\mathcal{H}^d| \leq k$*) minimal transversals of* $\mathcal{H}$*.*

This so called *hypergraph transversal problem* [3] has also received considerable attention in the literature, since it is known to be polynomially equivalent with many other problems appearing in various areas, such as artificial intelligence (e.g. [18, 25, 34]), database theory [44, 52], distributed systems [23, 30], game theory (e.g. [28, 48, 49]), machine learning and data mining (e.g. [2, 27]), mathematical programming (e.g. [7, 37]), matroid theory (e.g. [9, 38]), and reliability theory (e.g. [12, 48]).

It follows from [22] that problem GEN-TRANS$(\mathcal{H}, k)$ can be solved in *incremental quasi-polynomial* time $O(n(m+k)) + (m+k)^{o(\log(m+k))}$, where $m = |\mathcal{H}|$. For many special classes of hypergraphs, polynomial time algorithms are also known, e.g. hypergraphs of bounded edge size [5, 10, 14, 18, 21, 31], of bounded-degree [16, 19, 47], of bounded-edge intersections [6], of bounded treewidth [19], regular hypergraphs (e.g. [11, 13, 43, 46]), read-once [17], and several other classes of hypergraphs [15, 24, 39, 42]. From a practical point of view, our assymetric bounds stated above, when applied to generating hypergraph transversals, are of particular interest since typically the size of the output hypergraph $\mathcal{H}^d$ is much larger than the size of the input $|\mathcal{H}|$.

Note that the generation of the dual hypergraph $\mathcal{H}^d$ by repeated duality testing may require, in general, space that is exponential in n and the size of the input hypergraph $\mathcal{H}$. Tamaki [51] showed that a modification of the Fredman-Khachiyan algorithm can be implemented to generate all minimal transversals of $\mathcal{H}$ in *total polynomial* time $O(nm + K)^{O(\log(nm))}$, using *polynomial space* $O(nm\log(nm))$, independent of $K = |\mathcal{H}^d|$. As our second result (c.f. Theorem 2), we improve these bounds by showing that, for any $k \geq 1$, we can generate k minimal transversals of $\mathcal{H}$ in time $O(nk) + m^{o(\log k)}$ using space $o(n^2 m \log m)$ if $k >> n$, and in time $(n+k)^{O(\log m)}$ using space $O(nm \log m)$ in general. This result will be proved in Section 3. A more precise summary of these results is given in the next section.

1.1 Main results

Before stating our results, we need a few definitions which will use throughout the paper. For $v \in \mathbb{R}_+$, let $\chi(v)$ be the unique positive root of the equation $(\chi(v)/2)^{\chi(v)} = v$. For $a, b \in \mathbb{R}_+$, let $c = c(a,b)$ be the unique positive root of the equation

$$2^{c+1}\left(a^{c/\log b} - 1\right) = 1. \tag{1}$$

Given $n, K \in \mathbb{R}_+$, we define $\epsilon = \epsilon(n,K)$ and $\kappa = \kappa(n,K)$, as

$$\epsilon = \frac{1}{2}\left[\frac{1}{2} + \left(\frac{2K}{n}\right)^{\frac{1}{\kappa-1}}\right]^{-1}, \qquad \kappa = \frac{\log(2K/n)}{c(2n, 2K/n)} + 1, \tag{2}$$

if $2K > n$, and we set

$$\epsilon = \frac{1}{3}, \qquad \kappa = \frac{\log(2n)}{\log(3/2)} + 1, \tag{3}$$

otherwise. Note that $\kappa > 1$ for $n \geq 1$ and hence $\epsilon \leq 1/3$.

In stating our bounds below, we shall only give bounds in terms of the depth and number of nodes of the recursion tree used in the algorithm. It should be understood that these bounds on the parallel running time and the number of processors should be multiplied, respectively, by polylogarithmic and polynomial factors of n, m and K, corresponding to the operations performed at each recursion tree node.

Theorem 1 *Let $f, g : \{0,1\}^n \mapsto \{0,1\}$ be two monotone Boolean functions given by their irredundant disjunctive normal forms, represented by Sperner hypergraphs $\mathcal{F}, \mathcal{G} \subseteq 2^V$, of sizes $|\mathcal{F}| = m$ and $|\mathcal{G}| = K$. Then (i) Duality-testing(f, g) can be solved in time $2\chi(N)\ln(N) + 1$ using $N^{4\chi(N)}$ processors, where $N = m + K$. (ii) Duality-testing(f, g) can be solved in time*[1] *$\kappa(n,K)\log m/\log(2n) + 1$ using $m^{\kappa(n,K)}$ processors, and in time $\kappa(n,m)\log K/\log(2n) + 1$ using $K^{\kappa(n,m)}$ processors.*

Note that $\chi(N) \sim \log N/\log\log N$. Note also that if $K > n/2$, then $\kappa(n,K) \sim \log(2K/n)/\log\left(\frac{\log(2K/n)}{\log(2n)}\right)$.

For $x \in [0,1]$, let

$$H(x) = x \log \frac{1}{x} + (1-x)\log \frac{1}{1-x}. \tag{4}$$

With respect to generating hypergraph transversals we obtain the following result.

Theorem 2 *Let $\mathcal{H} \subseteq 2^V$ be a hypergraph, and $k \geq 1$ is a given integer. Then problem GEN-TRANS($\mathcal{H}, k$) can be solved in time $(n + k')^{O(\log m)}$ using space $O(nm\log m)$, and in time $O(n(k' + m^{\kappa(n,k')}\log k'))$ using space $O(nm(\log m + 1)/H(\epsilon(n,k')))$, where $k' = \min\{k, |\mathcal{H}^d|\}$.*

[1] We use $\log(\cdot)$ for the base 2-logarithm and $\ln(\cdot)$ for the natural logarithm.

If $k' > n/2$, we get by the definition of ϵ and κ that

$$H(\epsilon) = \frac{\epsilon(\log(4k') + \kappa - 1) + (1-\epsilon)\log(2n)}{\kappa - 1} \geq \frac{\log(2n)}{\kappa - 1},$$

while $H(\epsilon) = H(1/3)$ if $k' \leq n/2$. Thus we obtain from Theorem 2 that the required space is $O(nm\kappa(n,k')\log m/\log n)$ if $k' > n/2$ and $M(n,m) = O(nm\log m)$ otherwise. Since $k' \leq |\mathcal{H}^d| \leq 2^n$, and $\kappa(n, 2^n) = o(n)$, we get the space required is $O(n^2 m \log m/\log n)$.

1.2 Generating characterizing families of monotone properties

Let V be a finite set of cardinality n and $\pi : 2^V \mapsto \{0,1\}$ be a *monotone* property defined over the subsets of V. We assume that π is represented by a parallel *satisfiability oracle* $\mathcal{O}_\pi$, i.e. an algorithm $\mathcal{O}_\pi : \mathcal{D}_\pi \times 2^V \mapsto \{0,1\}$ which, given some input description $\mathcal{D}_\pi$ of π and a subset $X \subseteq V$, can decide whether or not $\pi(X) = 1$, in time polylogarithmic in $n = |V|$ and the size $|\mathcal{D}_\pi|$ of the input description of π. Since π is monotone, it may be also represented by either the family $\mathcal{F}_\pi \subseteq 2^V$ of *minimal sets satisfying* π, or by the family $\mathcal{G}_\pi \subseteq 2^V$ of *maximal sets not satisfying* π.

While the *separate* enumeration of either one the families $\mathcal{F}_\pi$ or $\mathcal{G}_\pi$, for a given monotone property π, may be an NP-hard problem (see e.g. [40]), it was independently observed by Bioch and Ibaraki [4] and Gurvich and Khachiyan [29] that the *joint* generation of $\mathcal{F}_\pi$ and $\mathcal{G}_\pi$, that is the problem:

JOINT-GEN$(\mathcal{F}_\pi, \mathcal{G}_\pi, \mathcal{X}, \mathcal{Y})$**:** *Given two explicitly listed set families $\mathcal{X} \subseteq \mathcal{F}_\pi$ and $\mathcal{Y} \subseteq \mathcal{G}_\pi$, find a new set in $(\mathcal{F}_\pi \setminus \mathcal{X}) \cup (\mathcal{G}_\pi \setminus \mathcal{Y})$, if one exists,*

can be reduced in polynomial time to the monotone Boolean dualization problem. One may further observe that such reductions are indeed NC-reductions to the duality testing problem and/or to the the generation of a *single minimal set* satisfying π. For the latter problem, Karp, Upfal, and Wigderson [33] gave a randomized algorithm which makes only $O(\sqrt{n})$ parallel oracle calls on $O(n^{3/2})$ processors. Combining this with Theorem 1, we arrive at the following fairly general result.

Theorem 3 *Let $\pi : 2^V \mapsto \{0,1\}$ be a monotone property defined by a parallel satisfiability oracle, and $\mathcal{X} \subseteq \mathcal{F}_\pi$ and $\mathcal{Y} \subseteq \mathcal{G}_\pi$ be given subfamilies. Then problem JOINT-GEN$(\mathcal{F}_\pi, \mathcal{G}_\pi, \mathcal{X}, \mathcal{Y})$ can be solved in $O(\sqrt{|V|}\,\mathrm{polylog}(|V|, |\mathcal{X}|, |\mathcal{Y}|, |\mathcal{D}_\pi|)$ randomized parallel time using quasi-poly$(|V|, |\mathcal{X}|, |\mathcal{Y}|, |\mathcal{D}_\pi|)$ number of processors.*

1.3 Notation and some preliminaries

In what follows, we let V be a finite set of n elements and represent Monotone Boolean functions f, g by the indices of their irredundant disjunctive normal forms, represented as Sperner hypergraphs $\mathcal{F}, \mathcal{G} \subseteq 2^V$ respectively. For a hypergraph $\mathcal{H} \subseteq 2^V$ and a subset

$S \subseteq V$, denote by $\mathcal{H}_S = \{H \in \mathcal{H} \mid H \subseteq S\}$ the sub-hypergraph of $\mathcal{H}$ induced by S, by $\mathcal{H}^S = \{\text{minimal } (H \cap S) \mid H \in \mathcal{H}\}$ the restriction of $\mathcal{H}$ into S, and by $\mathcal{H}(S) = \{H \in \mathcal{H} \mid H \cap S \neq \emptyset\}$, the hypergraph consisting of those hyperedges of $\mathcal{H}$ intersecting S. For a vertex $v \in V$, we denote by $\deg_{\mathcal{H}}(v)$ the degree of v in the hypergraph $\mathcal{H} \subseteq 2^V$, i.e. $\deg_{\mathcal{H}}(v) = |\{H \in \mathcal{H} \ : \ v \in H\}|$. For $S \subseteq V$, we use the notation $\bar{S} = V \setminus S$. A hypergraph $\mathcal{H}$ is said to be trivial if $\mathcal{H} = \emptyset$ or $\mathcal{H} = \{\emptyset\}$.

For a hypergraph $\mathcal{H} \subseteq 2^V$ and a positive number $\epsilon \in (0,1)$, denote by $L = L(\mathcal{H}, \epsilon)$ the vertex subset $\{v \in V : \ \deg_{\mathcal{H}}(v) \leq \epsilon|\mathcal{H}|\}$. 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 $\epsilon''|\mathcal{H}| \leq |\mathcal{H}(S)| \leq \epsilon'|\mathcal{H}|$. If $\epsilon_1 < \epsilon_2$ are two given numbers in $(0,1)$, and $L = L(\mathcal{H}, \epsilon_1)$ satisfies $|\mathcal{H}(L)| > \epsilon_2|\mathcal{H}|$, then there exists an $(\epsilon_2, \epsilon_2 - \epsilon_1)$-balanced subset of L, which we will denote by L', and simply call henceforth a balanced subset of L. The set L' can be found efficiently *in parallel* (i.e. in NC time) as follows. Let $\{v_1, \ldots, v_l\}$ be an arbitrary order of the vertices of L and find the index $j \in [l-1]$, such that

$$|\mathcal{H}(\{v_1, \ldots, v_j\})| \leq \epsilon_2|\mathcal{H}| \text{ and } |\mathcal{H}(\{v_1, \ldots, v_{j+1}\})| > \epsilon_2|\mathcal{H}|. \tag{5}$$

The existence of such index j is guaranteed by the facts that $|\mathcal{H}(\{v_1\})| \leq \epsilon_1|\mathcal{H}| < \epsilon_2|\mathcal{H}| < |\mathcal{H}(L)|$. Finally, we let $L' = \{v_1, \ldots, v_j\}$. Since $\deg_{\mathcal{H}}(v_{j+1}) \leq \epsilon_1|\mathcal{H}|$, it follows from (5) that $|\mathcal{H}(L')| > (\epsilon_2 - \epsilon_1)|\mathcal{H}|$, implying that L' is indeed a balanced subset of L.

2 Duality testing

2.1 Symmetric bounds

Let $f, g : \{0,1\}^n \mapsto \{0,1\}$ be two monotone Boolean functions represented by Sperner hypergraphs $\mathcal{F}, \mathcal{G} \subseteq 2^V$. A *witness* for the non-duality of the pair (f, g) is a Boolean vector $x \in \{0,1\}^n$, such that $g(x) = f(\bar{x})$. Clearly, if $F \cap G = \emptyset$ for some $F \in \mathcal{F}$ and $G \in \mathcal{G}$, then the characteristic vector of $\bar{F}$ is a witness for the non-duality of (f, g). Thus we may assume throughout that

$$F \cap G \neq \emptyset \text{ for all } F \in \mathcal{F} \text{ and } G \in \mathcal{G}. \tag{6}$$

We shall say that the hypergraphs $\mathcal{F}$ and $\mathcal{G}$ are dual if and only if there exists no $X \subseteq V$ such that

$$X \cap F \neq \emptyset \text{ for all } F \in \mathcal{F}, \text{ and } X \not\supseteq G \text{ for all } G \in \mathcal{G}. \tag{7}$$

Also, by definition, the pair $(\emptyset, \{\emptyset\})$ is dual. Note that condition (7) is symmetric in $\mathcal{F}$ and $\mathcal{G}$: $X \subseteq V$ satisfies (7) for the pair $(\mathcal{F}, \mathcal{G})$ if and only if $\bar{X}$ satisfies (7) for $(\mathcal{G}, \mathcal{F})$. Clearly, $\mathcal{F}^d = \mathcal{G}$, and consequently $f^d = g$, if and only if $(\mathcal{F}, \mathcal{G})$ form a dual pair, and conversely, the existence of an $X \subseteq V$ satisfying (7) provides a witness for the non-duality of the pair (f, g).

Proposition 1 ([22]) *Let $\mathcal{F}, \mathcal{G} \subseteq 2^V$ be two hypergraphs satisfying (6), and $v \in V$ be a given vertex. Then $\mathcal{F}$ and $\mathcal{G}$ are dual if and only if the pairs $(\mathcal{F}_{V \setminus v}, \mathcal{G}^{V \setminus v})$ and $(\mathcal{F}^{V \setminus v}, \mathcal{G}_{V \setminus v})$ are dual.*

Based on Proposition 1, the following was proposed in [22]: assuming the duality of $\mathcal{F}$ and $\mathcal{G}$, there exists a vertex $u \in V$ with "large" degree in at least one of the two families. Using this variable, the problem can be decomposed into two subproblems, one with size at most $(1 - 1/\log v)v$ and the other of size $v-1$, where $v = |\mathcal{F}||\mathcal{G}|$ is the measure taken for the size of the problem. This leads to a simple recursive algorithm A whose running time is $v^{O(\log^2 v)}$. A more complicated algorithm B was then presented, running in time $v^{o(\log v)}$, that makes use of the dependence between the two subproblems arising after applying Proposition 1. We note that in both algorithms A and B, if we consider any node corresponding to a subproblem of size v, then one of the children of this node may have size $v-1$. This implies that the recursion tree depth can be linear in the problem size, a property that cannot lead to efficient parallel algorithms. In order to obtain a tree with polylogarithmic depth, we perform decomposition on a suitable subset of vertices, rather than a single vertex.

Proposition 2 *Let $\mathcal{F}, \mathcal{G} \subseteq 2^V$ be two hypergraphs satisfying (6), and $L \subseteq V$ be a given subset of vertices. Then $\mathcal{F}$ and $\mathcal{G}$ are dual if and only if*

$$\mathcal{F}_{\bar{L}} \text{ and } \mathcal{G}^{\bar{L}} \text{ are dual,} \tag{8}$$

$$\mathcal{F}_{\bar{Y}} \text{ and } \mathcal{G}^{\bar{Y}} \text{ are dual for all } Y \in \mathcal{G}^{\bar{L}}. \tag{9}$$

Proof. Suppose that $\mathcal{F}$ and $\mathcal{G}$ are dual. If (8) is not satisfied, then there exists an $X \subseteq \bar{L}$, intersecting every edge of $\mathcal{F}_{\bar{L}}$, and containing no edge of $\mathcal{G}^{\bar{L}}$. Note that, by (6), $L \not\supseteq G$ for all $G \in \mathcal{G}$ (otherwise, $\mathcal{F}_{\bar{L}} = \emptyset$, $\mathcal{G}^{\bar{L}} = \{\emptyset\}$, and we get that $\mathcal{F}_{\bar{L}}$ and $\mathcal{G}^{\bar{L}}$ are dual). Thus $X \cup L$ satisfies (7) for the pair $(\mathcal{F}, \mathcal{G})$, in contradiction to their assumed duality. On the other hand, if (9) does not hold for some $Y \in G^{\bar{L}}$, then there is an $X \subseteq \bar{Y}$, satisfying (7) for the pair $(\mathcal{F}_{\bar{Y}}, \mathcal{G}^{\bar{Y}})$. Thus $X \cup Y$ intersects every edge of $\mathcal{F}$. Furthermore, $X \cup Y$ does not contain any edge in $\mathcal{G}$ (note that if Y contains some edge $G \in \mathcal{G}$, then by (6), $\mathcal{F}_{\bar{Y}} = \emptyset$, and $\mathcal{G}^{\bar{Y}} = \{\emptyset\}$, and hence (9) is satisfied). This again contradicts the fact that $\mathcal{F}$ and $\mathcal{G}$ are dual.

Conversely, suppose that $\mathcal{F}$ and $\mathcal{G}$ are not dual, i.e. there exists an $X \subseteq V$ satisfying (7) for the pair $(\mathcal{F}, \mathcal{G})$. If $X \not\supseteq Y$ for all $Y \in \mathcal{G}^{\bar{L}}$, then $X \cap \bar{L}$ is an evidence that (8) is not satisfied. Otherwise, let $G \in \mathcal{G}$ be such that $Y = G \cap \bar{L} \subseteq X$. Then $X \cap \bar{Y}$ intersects every edge $F \in \mathcal{F}_{\bar{Y}}$ and does not contain any edge in $\mathcal{G}^{\bar{Y}}$, i.e. (9) is not satisfied. $\Box$

Observe that all pairs of hypergraphs $(\mathcal{F}, \mathcal{G})$ arising in (8) and (9) satisfy condition (6). Based on Proposition 2, we can prove part (i) of Theorem 1 using the following algorithm.

Algorithm P1. Consider the following parallel procedure DAUL$(V, \mathcal{F}, \mathcal{G})$ for testing the duality of a pair of hypergraphs $\mathcal{F}, \mathcal{G} \subseteq 2^V$, satisfying (6). The procedure returns True or False depending on the duality/non-duality of $\mathcal{F}$ and $\mathcal{G}$. It can be easily modified to return a witness $X \subseteq V$ in the case of non-duality. If $\min\{|\mathcal{F}|, |\mathcal{G}|\} \leq 1$, then duality of $\mathcal{F}$ and $\mathcal{G}$ can be tested efficiently in parallel using a simple procedure which we call SIMPLE-DUAL.

Procedure DAUL$(V,\mathcal{F},\mathcal{G})$**:**

1. **if** $\min\{|\mathcal{F}|,|\mathcal{G}|\} \le 1$, **then return** SIMPLE-DUAL$(V,\mathcal{F},\mathcal{G})$.
2. $v{:=}\ |\mathcal{F}||\mathcal{G}|$, $\epsilon_1{:=}\ \frac{1}{\chi(v)}$, $\epsilon_2{:=}\ \frac{2}{\chi(v)}$, $L_1{:=}\ L(\mathcal{F},\epsilon_1)$ and $L_2{:=}\ L(\mathcal{G},\epsilon_1)$.
3. **if** $|\mathcal{F}(L_1)| < |\mathcal{F}|$ and $|\mathcal{G}(L_2)| < |\mathcal{G}|$, **then** (by (6), $\bar{L}_1 \cap \bar{L}_2 \neq \emptyset$)
   $v{:=}$ a vertex in $\bar{L}_1 \cap \bar{L}_2$;
   in parallel, **do** recursively (c.f. Proposition 2)
   $d_1{:=}$ DUAL$(V \setminus v, \mathcal{F}_{V\setminus v}, \mathcal{G}^{V\setminus v})$, $d_2{:=}$ DUAL$(V \setminus v, \mathcal{F}^{V\setminus v}, \mathcal{G}_{V\setminus v})$;
   return $d_1 \wedge d_2$.
4. **if** $|\mathcal{F}(L_1)| = |\mathcal{F}|$, **then**
   $L_1'{:=}$ a balanced subset of L_1;
   in parallel, **do** recursively (c.f. Proposition 2)
   $d_1{:=}$ DUAL$(\bar{L}_1', \mathcal{F}_{\bar{L}_1'}, \mathcal{G}^{\bar{L}_1'})$ and $d_Y{:=}$ DUAL$(\bar{Y}, \mathcal{F}_{\bar{Y}}, \mathcal{G}^{\bar{Y}})$ **forall** $Y \in \mathcal{G}^{\bar{L}_1'}$;
   return $d_1 \wedge (\bigwedge_{Y\in\mathcal{G}^{\bar{L}_1'}} d_Y)$.
5. **otherwise** $(|\mathcal{G}(L_2)| = |\mathcal{G}|)$,
   $L_2'{:=}$ a balanced subset of L_2;
   in parallel, **do** recursively
   $d_1{:=}$ DUAL$(\bar{L}_2', \mathcal{F}^{\bar{L}_2'}, \mathcal{G}_{\bar{L}_2'})$ and $d_Y{:=}$ DUAL$(\bar{Y}, \mathcal{F}^{\bar{Y}}, \mathcal{G}_{\bar{Y}})$ **forall** $Y \in \mathcal{F}^{\bar{L}_2'}$;
   return $d_1 \wedge (\bigwedge_{Y\in\mathcal{F}^{\bar{L}_2'}} d_Y)$.

Analysis of P1. Let respectively $T(v)$ and $P(v)$ be the parallel time and number of processors (in units each of which, respectively, corresponds to the time and number of processors used by a single recursive call) used on an instance of size $v = |\mathcal{F}||\mathcal{G}|$.

Step 3: Since $v \in \bar{L}_1 \cap \bar{L}_2$, we have $\deg_{\mathcal{F}}(v) > \epsilon_1|\mathcal{F}|$ and $\deg_{\mathcal{G}}(v) > \epsilon_1|\mathcal{G}|$. Thus $|\mathcal{F}_{V\setminus v}| < (1-\epsilon_1)|\mathcal{F}|$, and $|\mathcal{G}_{V\setminus v}| < (1-\epsilon_1)|\mathcal{G}|$, and we get the recurrences

$$T(v) \le 1 + T((1-\epsilon_1)v), \tag{10}$$

$$P(v) \le 1 + 2P((1-\epsilon_1)v). \tag{11}$$

Step 4: Let $|\mathcal{F}(L')| = \epsilon|\mathcal{F}|$. Then $|\mathcal{F}_{\bar{L}'}| \le (1-\epsilon)|\mathcal{F}|$ and $|\mathcal{F}_{\bar{Y}}| \le \epsilon|\mathcal{F}|$ (since $\mathcal{F}_{\bar{Y}} \subseteq \mathcal{F}(L')$ by (6)) for all $Y \in \mathcal{G}^{\bar{L}'}$. Since $|\mathcal{G}^{\bar{L}_1'}| \le |\mathcal{G}| \le v/2$, assuming $|\mathcal{F}| > 1$, we get the recurrences

$$T(v) \le 1 + \max\{T((1-\epsilon)v), T(\epsilon v)\}, \tag{12}$$

$$P(v) \le 1 + P((1-\epsilon)v) + \frac{v}{2}P(\epsilon v), \tag{13}$$

Furthermore, we know that $\epsilon \in (\epsilon_2 - \epsilon_1, \epsilon_2]$, since L_1' is a balanced subset of L_1.

Step 5: By symmetry we get the same recurrences (12) and (13).

Following [22], we prove by induction on $v \geq 1$ that $T(v) \leq \chi(v)\ln v+1$ and $P(v) \leq v^{\chi(v)}$. Consider (10) and (11). By induction and monotonicity of $\chi(\cdot)$, we get from (10)

$$\begin{aligned} T(v) &\leq 1+\chi((1-\epsilon_1)v)\ln((1-\epsilon_1)v)+1 \leq 1+\chi(v)\ln\left(\left(1-\frac{1}{\chi(v)}\right)v\right)+1 \\ &= 1+\chi(v)\ln v+\chi(v)\ln\left(1-\frac{1}{\chi(v)}\right)+1 \leq \chi(v)\ln v+1, \end{aligned}$$

where we used the inequality $1-x \leq e^{-x}$, valid for all real x. On the other hand, (11) gives

$$\begin{aligned} P(v) &\leq 1+2((1-\epsilon_1)v)^{\chi(v)} \leq 1+2\left(1-\frac{1}{\chi(v)}\right)^{\chi(v)} v^{\chi(v)} \\ &\leq 1+\frac{2}{e}v^{\chi(v)} \leq v^{\chi(v)}, \end{aligned}$$

for v large enough ($v \geq 4$ is sufficient, explaining our stopping criterion in Step 1 of the algorithm).

Now consider recurrence (12). Applying induction, we get

$$T(v) \leq 1+\chi(v)\max\left\{\ln\left(\left(1-\frac{1}{\chi(v)}\right)v\right), \ln\left(\frac{2v}{\chi(v)}\right)\right\}+1 \leq \chi(v)\ln v+1,$$

since $\chi(v) \geq 3$ for $v \geq 4$. On the other hand, recurrence (13) gives

$$\begin{aligned} P(v) &\leq 1+P\left(\left(1-\frac{1}{\chi(v)}\right)v\right)+\frac{v}{2}P\left(\frac{2v}{\chi(v)}\right) \\ &\leq 1+\left[\left(1-\frac{1}{\chi(v)}\right)^{\chi(v)}+\frac{v}{2}\left(\frac{2}{\chi(v)}\right)^{\chi(v)}\right]v^{\chi(v)} \\ &\leq 1+\left[\frac{1}{e}+\frac{1}{2}\right]v^{\chi(v)} < v^{\chi(v)}, \end{aligned}$$

for $v \geq 4$. □

2.2 Asymmetric bounds

Note that, in algorithm P1, we used the symmetry between the families $\mathcal{F}$ and $\mathcal{G}$, in a way similar to that used in [22]. In order to obtain assymetric bounds, we need a decomposition rule that does not use such symmetry. This is provided by the next proposition.

Proposition 3 *Let $\mathcal{F}, \mathcal{G} \subseteq 2^V$ be two hypergraphs on vertex set V, satisfying condition (6). Then $\mathcal{F}$ and $\mathcal{G}$ are dual if and only if*

$$\mathcal{F}_{V\setminus v} \text{ and } \mathcal{G}^{V\setminus v} \text{ are dual for all } v \in V. \tag{14}$$

Proof. Suppose that there exists $v \in V$ such that $\mathcal{F}_{V\setminus v}$ and $\mathcal{G}^{V\setminus v}$ are not dual. Note that $\{v\} \notin \mathcal{G}$, for otherwise (6) implies that $\mathcal{F}_{V\setminus v} = \emptyset$, and hence $(\mathcal{F}_{V\setminus v}, \mathcal{G}^{V\setminus v}) = (\emptyset, \{\emptyset\})$ form a dual pair. Then there exists an $X \subseteq V \setminus v$, such that X intersects every edge of $\mathcal{F}_{V\setminus v}$ and contains no edge of $\mathcal{G}^{V\setminus v}$. This gives a set $X \cup \{v\}$ which intersects every edge of $\mathcal{F}$ and contains no edge of $\mathcal{G}$, thus disproving the duality of $\mathcal{F}$ and $\mathcal{G}$.

Conversely, if X satisfies (7) for $(\mathcal{F}, \mathcal{G})$, then for every $v \in V$, $X \setminus v$ intersects every edge in $\mathcal{F}_{V\setminus v}$. Furthermore, if for some $v \in V$, $X \setminus v$ contains some edge in $Y \in \mathcal{G}^{V\setminus v}$, then $G = Y \cup \{v\} \in \mathcal{G}$ and hence X cannot contain v. It follows that, unless X is empty, there must exist a $v \in V$, such that X does not contain any edge in $\mathcal{G}^{V\setminus v}$. This disproves the duality of $(\mathcal{F}_{V\setminus v}, \mathcal{G}^{V\setminus v})$. $\square$

Combining Propositions 2 and 3, we obtain the following decomposition rule.

Corollary 1 *Let $\mathcal{F}, \mathcal{G} \subseteq 2^V$ be two hypergraphs satisfying (6), and $L \subseteq V$ be a given subset of vertices. Then $\mathcal{F}$ and $\mathcal{G}$ are dual if and only if*

$$\mathcal{F}_{\bar{L}\setminus v} \text{ and } \mathcal{G}^{\bar{L}\setminus v} \text{ are dual, for all } v \in \bar{L},$$
$$\mathcal{F}_{\bar{Y}} \text{ and } \mathcal{G}^{\bar{Y}} \text{ are dual for all } Y \in \mathcal{G}^{\bar{L}}.$$

For the next procedure, we use the thresholds $\epsilon_2 = 2\epsilon_1 = 2\epsilon$, where ϵ is given by (2) and (3). Note that the values of the thresholds are computed with respect to the *initial* values of $n = |V|$ and $K = |\mathcal{G}|$.

Algorithm P2. Consider the following parallel procedure:

Procedure DAUL$(V, \mathcal{F}, \mathcal{G})$**:**

1. **if** $|\mathcal{F}| \leq 1$, **then return** SIMPLE-DUAL$(V, \mathcal{F}, \mathcal{G})$.
2. $L := L(\mathcal{F}, \epsilon_1)$.
3. **if** $|\mathcal{F}(L)| \leq \epsilon_2 |\mathcal{F}|$, **then** (c.f. Corollary 1)
   in parallel, **do** recursively
      $d_v :=$ DUAL$(\bar{L} \setminus v, \mathcal{F}_{\bar{L}\setminus v}, \mathcal{G}^{\bar{L}\setminus v})$ **forall** $v \in \bar{L}$;
      $d_Y :=$ DUAL$(\bar{Y}, \mathcal{F}_{\bar{Y}}, \mathcal{G}^{\bar{Y}})$ **forall** $Y \in \mathcal{G}^{\bar{L}}$;
   return $(\bigwedge_{v\in\bar{L}} d_v) \wedge (\bigwedge_{Y\in\mathcal{G}^{\bar{L}}} d_Y)$.
4. **otherwise** $(|\mathcal{F}(L)| > \epsilon_2 |\mathcal{F}|)$,
   $L' :=$ a balanced subset of L;
   in parallel, **do** recursively (c.f. Proposition 2)
      $d_1 :=$ DUAL$(\bar{L'}, \mathcal{F}_{\bar{L'}}, \mathcal{G}^{\bar{L'}})$ and $d_Y :=$ DUAL$(\bar{Y}, \mathcal{F}_{\bar{Y}}, \mathcal{G}^{\bar{Y}})$ **forall** $Y \in \mathcal{G}^{\bar{L'}}$;
   return $d_1 \wedge (\bigwedge_{Y\in\mathcal{G}^{\bar{L'}}} d_Y)$.

Analysis of P2. Let respectively $T(n, m, K)$ and $P(n, m, K)$ be the parallel time and number of processors used on an instance of the problem of size (n, m, K), where $n = |V|$,

$m = |\mathcal{F}|$, and $K = |\mathcal{G}|$. We assume for this analysis that the values of n and K are "frozen" at their initial values, and thus we can express the running time and number of processors as functions $T(m)$ and $P(m)$ of only one variable m.

Step 3: Note that, for all $v \in \bar{L}$, $\deg_{\mathcal{F}}(v) > \epsilon_1|\mathcal{F}|$. Thus $|\mathcal{F}_{\bar{L}\setminus v}| \leq (1-\epsilon_1)|\mathcal{F}|$. Since $|\mathcal{F}(L)| \leq \epsilon_2|\mathcal{F}|$, we get the recurrences

$$\begin{aligned} T(m) &\leq 1 + \max\{T((1-\epsilon_1)m), T(\epsilon_2 m)\} && (15)\\ P(m) &\leq 1 + n \cdot P((1-\epsilon_1)m) + K \cdot P(\epsilon_2 m). && (16)\end{aligned}$$

Step 4: Note that, by our selection of L', we have $(\epsilon_2 - \epsilon_1)|\mathcal{F}| < |\mathcal{F}(L')| \leq \epsilon_2|\mathcal{F}|$. Hence, we get the recurrences

$$\begin{aligned} T(m) &\leq 1 + \max\{T((1-(\epsilon_2-\epsilon_1))m), T(\epsilon_2 m)\} && (17)\\ P(m) &\leq 1 + P((1-(\epsilon_2-\epsilon_1))m) + K \cdot P(\epsilon_2 m). && (18)\end{aligned}$$

We prove by induction on $m \geq 1$ that $T(m) \leq \kappa(n,K)\log m/\log(2n) + 1$ and $P(m) \leq m^{\kappa(n,K)}$. (A similar bound for $P(m)$ can be obtained using the Akra-Bazzi method [1].) Clearly it is enough to prove this for (15) and (16).

Case 1. $n < 2K$: First we may verify that our setting (2) together with (1) imply that

$$n(1-\epsilon)^{\kappa} + K(2\epsilon)^{\kappa} = n(1-\epsilon)^{\kappa-1} = 2K(2\epsilon)^{\kappa-1} = \frac{1}{2}. \tag{19}$$

Now consider (15): we get by induction and (19) that

$$\begin{aligned} T(m) &\leq 1 + \frac{\kappa \ln m}{\log(2n)} + 1 + \kappa \cdot \max\left\{\frac{\log(1-\epsilon)}{\log(2n)}, \frac{\log(2\epsilon)}{\log(2n)}\right\}\\ &= 1 + \frac{\kappa \ln m}{\log(2n)} + 1 + \kappa \cdot \max \cdot \left\{\frac{-1}{\kappa-1}, \frac{-1}{\kappa-1}\left(\frac{\log(4K)}{\log(2n)}\right)\right\}\\ &= 1 + \frac{\kappa \ln m}{\log(2n)} + 1 - \frac{\kappa}{\kappa-1} < \frac{\kappa \log m}{\log(2n)} + 1.\end{aligned}$$

This gives the required bound on $T(m)$. Next, we consider (16) and apply induction and use (19) to get

$$P(m) \leq 1 + [n(1-\epsilon)^{\kappa} + K(2\epsilon)^{\kappa}]\, m^{\kappa} = 1 + \frac{1}{2}m^{\kappa} \leq m^{\kappa},$$

for $m \geq 2$. (In fact ϵ is selected to minimize $n(1-\epsilon)^{\kappa} + K(2\epsilon)^{\kappa}$.)

Case 2. $n \geq 2K$: First consider (15). Form (3), we get that $(3/2)^{\kappa-1} = 2n$. Both terms of the maximum in (15) are equal to $T(\frac{2}{3}m)$ and hence we get by induction

$$T(m) \leq 1 + \frac{\kappa \ln m}{\ln(2n)} + 1 + \frac{\kappa \ln(2/3)}{\log(2n)} = 1 + \frac{\kappa \log m}{\log(2n)} + 1 - \frac{\kappa}{\kappa-1} < \frac{\kappa \ln m}{\log(2n)} + 1.$$

Now consider (16). Using $\epsilon = 1/3$, $K \le n/2$, and induction, we get for $m \ge 2$

$$P(m) \;\le\; 1 + \frac{3n}{2}P(\frac{2}{3}m) \le 1 + \frac{3n}{2}\left(\frac{2}{3}\right)^{\kappa} m^{\kappa} = 1 + \frac{1}{2}m^{\kappa} \le m^{\kappa}.$$

By symmetry, if $m > n/2$, one can also test duality in parallel time $T(n,m,K) \le \kappa(n,m)\log K/\log(2n) + 1$ using $P(n,m,K) \le K^{\kappa(n,m)}$ processors (per recursive call). This establishes the bounds stated in part (ii) of Theorem 1. □

3 Polynomial space generation of minimal hypergraph transversals

3.1 Generating hypergraph transversals

Given two hypergraphs $\mathcal{H}_1, \mathcal{H}_2 \subseteq 2^V$, denote by

$$\begin{aligned}\mathcal{H}_1 \wedge \mathcal{H}_2 &= \{\text{minimal } H \mid H = H_1 \cup H_2 \text{ for some } H_1 \in \mathcal{H}_1 \text{ and } H_2 \in \mathcal{H}_2\}\\ \mathcal{H}_1 \vee \mathcal{H}_2 &= \{\text{minimal } H \mid H \in \mathcal{H}_1 \cup \mathcal{H}_2\}\end{aligned}$$

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

The following proposition is implicit in [22, 51].

Proposition 4 *Let $\mathcal{H}, \mathcal{F}, \mathcal{G} \subseteq 2^V$ be hypergraphs such that $\mathcal{H} = \mathcal{F} \vee \mathcal{G}$. Then*

$$\mathcal{H}^d = \bigvee_{Y \in \mathcal{F}^d} (\mathcal{G}^d_{\bar{Y}} \wedge \{Y\}). \tag{20}$$

Proof. Let $T \in \mathcal{H}^d$ be a minimal transversal to $\mathcal{H}$. Then T is a transversal to $\mathcal{F}$, and hence there exists a subset $Y \subseteq T$, such that $Y \in \mathcal{F}^d$. Also, T is a transversal to $\mathcal{G}$, and consequently $T \setminus Y$ must be a transversal to every set in $\mathcal{G}$ that does not intersect Y. Thus $T \setminus Y$ contains a set $X \in \mathcal{G}^d_{\bar{Y}}$, i.e. T contains a set $X \cup Y$ belonging to the right hand side of (20). Conversely, any set $X \cup Y$, such that $Y \in \mathcal{F}^d$ and $X \in \mathcal{G}^d_{\bar{Y}}$, must hit every set in $\mathcal{H} = \mathcal{F} \vee \mathcal{G}$, and thus must contain a minimal transversal to $\mathcal{H}$. The statement follows since both sides of (20) are Sperner, by definition. □

Proposition 5 *Let $\mathcal{H} \subseteq 2^V$ be a non-trivial hypergraph. Then*

$$\mathcal{H}^d = \bigvee_{v \in V} (\mathcal{H}^d_{V \setminus v} \wedge \{\{v\}\}). \tag{21}$$

Proof. Let $T \in \mathcal{H}^d$ be a minimal transversal to $\mathcal{H}$ and $v \in T$ be any vertex in T. Then $T \setminus \{v\} \in \mathcal{H}^d_{V \setminus v}$, i.e. $T \in \mathcal{H}^d_{V \setminus v} \wedge \{\{v\}\}$, and hence T contains a set X belonging to the right-hand side of (21). Conversely, let X be an arbitrary such set, then there exists a $v \in V$, such that $X \in \mathcal{H}^d_{V \setminus v} \wedge \{\{v\}\}$, i.e. $X = T \cup \{v\}$ for some $T \in \mathcal{H}^d_{V \setminus v}$. Thus X is a transversal of $\mathcal{H}$, i.e. it contains a set $T' \in \mathcal{H}^d$. The statement follows since both sides of (21) are Sperner. □

Corollary 2 *Let $\mathcal{H} \subseteq 2^V$ be a non-trivial hypergraph and $L \subseteq V$ be a subset of its vertices. Then*

$$\begin{aligned} \mathcal{H}^d_{\bar{L}} &= \bigvee_{v \in \bar{L}} (\mathcal{H}^d_{\bar{L} \setminus v} \wedge \{\{v\}\}), \\ \mathcal{H}^d &= \bigvee_{Y \in \mathcal{H}^d_{\bar{L}}} (\mathcal{H}^d_{\bar{Y}} \wedge \{Y\}). \end{aligned}$$

Proof. Since $\mathcal{H} = \mathcal{H}_{\bar{L}} \vee \mathcal{H}(L)$, we can combine Propositions (4) and (5), to compute $\mathcal{H}^d$. □

The following proposition is needed to insure that the subproblems resulting after applying Corollary 2 do not have much larger output-sizes than the original hypergraph.

Proposition 6 ([40]) *Let $\mathcal{H} \subseteq 2^V$ be a hypergraph, then (i) for any $S \subseteq V$, $|\mathcal{H}^d_S| \leq |\mathcal{H}|$, and in particular, (ii) if $\mathcal{F}, \mathcal{G} \subseteq 2^V$ such that $\mathcal{H} = \mathcal{F} \vee \mathcal{G}$, and $Y \in \mathcal{F}^d$, then $|\mathcal{G}^d_{\bar{Y}}| \leq |\mathcal{H}^d|$.*

Algorithm P3. Given a Sperner hypergraph $\mathcal{H} \subseteq 2^V$ and an integer $k \geq 1$, we let $\epsilon = \epsilon(n,k)$ and $\kappa = \kappa(n,k)$ be as given by (2) if $2k > n$, and by (3) if $2k \leq n$, and fix $\epsilon_2 = 2\epsilon_1 = 2\epsilon$. We shall use a subroutine EXTEND$(\mathcal{H}, S, \mathcal{G}, k)$ which, given a hypergraph $\mathcal{H} \subseteq V$, a subset of vertices $S \subseteq V$, a subset of minimal sub-transversals $\mathcal{G} \subseteq \mathcal{H}^d_{\bar{S}}$, and an integer k, extends the elements of $\mathcal{G}$ into minimal transversals of $\mathcal{H}$, and returns k of them. (This can be done as follows: (i) for each $Y \in \mathcal{G}$, compute a minimal transversal T_Y of $\mathcal{H}^S_{\bar{Y}} = \{H \cap S \mid H \cap Y = \emptyset\}$; (ii) return $\{Y \cup T_Y \mid Y \in \mathcal{G}\}$.) To generate k (or all of $k \leq |\mathcal{H}^d|$) elements of $\mathcal{H}^d$, we call the following procedure $GEN(\mathcal{H}, V, k)$:

Procedure GEN$(\mathcal{H}, V, k)$:

1. **if** $|\mathcal{H}| = 0$, **then return** $\{\emptyset\}$; **if** $\mathcal{H} = \{H\}$, **then** return $\{\{i\} \ : \ i \in H\}$.
2. L:= $L(\mathcal{H}, \epsilon_1)$.
3. **if** $|\mathcal{H}(L)| \leq \epsilon_2 |\mathcal{H}|$, **then** (c.f. Corollary 2)
    foreach $v \in \bar{L}$
        $\mathcal{G}[v]$:= GEN$(\mathcal{H}_{\bar{L} \setminus v}, \bar{L} \setminus v, k)$;
        if $|\mathcal{G}[v]| \geq k$, **then return** EXTEND$(\mathcal{H}, L \cup \{v\}, \mathcal{G}[v], k)$.
    $\mathcal{G}[\bar{L}]$:= $\bigvee_{v \in \bar{L}} (\mathcal{G}[v] \wedge \{\{v\}\})$;

if $|\mathcal{G}[\bar{L}]| \geq k$, **then return** EXTEND($\mathcal{H}, L, \mathcal{G}[\bar{L}], k$).
foreach $Y \in \mathcal{G}[\bar{L}]$
$\mathcal{G}[Y]$:= GEN($\mathcal{H}_{\bar{Y}}, \bar{Y}, k$);
if $|\mathcal{G}[Y]| \geq k$, **then** Return EXTEND($\mathcal{H}, Y, \mathcal{G}[Y], k$).
return (at most) k elements of the disjunction $\bigvee_{Y \in \mathcal{G}[\bar{L}]}(\mathcal{G}[Y] \wedge \{Y\})$.

4. **otherwise** ($|\mathcal{H}(L)| > \epsilon_2 |\mathcal{H}|$),
L':= a balanced subset of L;
$\mathcal{G}[\bar{L}']$:= GEN($\mathcal{H}_{\bar{L}'}, \bar{L}', k$);
if $|\mathcal{G}[\bar{L}']| = k$, **then return** EXTEND($\mathcal{H}, L', \mathcal{G}[\bar{L}'], k$).
foreach $Y \in \mathcal{G}[\bar{L}']$,
$\mathcal{G}[Y]$:= GEN($\mathcal{H}_{\bar{Y}}, \bar{Y}, k$);
if $|\mathcal{G}[Y]| \geq k$, **then return** EXTEND($\mathcal{H}, Y, \mathcal{G}[Y], k$).
return (at most) k elements of the disjunction $\bigvee_{Y \in \mathcal{G}[\bar{L}']}(\mathcal{G}[Y] \wedge \{Y\})$.

Let $C(n, m, k)$ be the total number of recursive calls made by procedure GEN($\mathcal{H}, V, k$) on a problem of size (n, m, k), where $n = |V|$ and $m = |\mathcal{H}|$. With $C(n, m, k) = 1$ for $m \leq 1$, it follows exactly as in the analysis of P2 that $C(n, m, k) \leq m^{\kappa(n,k)}$. On the other hand, if we replace in P3, $\epsilon_1 < \epsilon_2$ by arbitrary constants in $(0, 1)$, say $\epsilon_1 = 1/3$ and $\epsilon_2 = 2/3$, then we get $C(n, m, k) \leq (n + k)^{O(\log m)}$.

Note that the way procedure P3 is stated requires that the computation of the transversal hypergraphs (e.g. $\mathcal{G}[v]$, $\mathcal{G}[\bar{L}]$, $\mathcal{G}[Y]$, etc) 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 that node. This is in fact unnecessary and can be avoided with a more careful implementation as will be seen in the next section.

3.2 Implementation with polynomial space

First, if $|\mathcal{H}^d| < k$, we would like to modify Algorithm P3 such that the running time depends on $|\mathcal{H}^d|$ and not on k. This can be done easily as follows. Let k' be the smallest power of 2 which is at least k. For each value of $k'' = 1, 2, 4, \ldots, k'$, we run procedure P3 with input k'', but we do not output the minimal transversals yet. This way, we determine an integer $k'' \leq 2 \min\{k, |\mathcal{H}^d|\}$. Now, we run procedure P3 with $k = k''$ and output at most k'' minimal transversals of $\mathcal{H}$ involving in total a number of $m^{\kappa(n,k'')} \log k''$ recursive calls. In a similar way, we can estimate the value of $K = |\mathcal{H}^d|$ within a factor of 2, by running P3 for each value of $k = 1, 2, 4, \ldots$ For each such value k, if the procedure did not stop after $m^{\kappa(n,k)} + 1$ recursive calls, we know that $K > k$, and we consider the next k, etc.

Next, we describe how to implement Procedure P3 with polynomial space. We use similar techniques as in [51]. Consider the recursion tree $\mathbf{T}$ (of depth at most $\kappa(n, k) \log m / \log(2n) + 1$, and number of nodes of at most $m^{\kappa(n,k)}$). An execution of Procedure P3 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 four types, say 0, 1, 2, or 3, corresponding to which case of procedure P3 applies at that node: Type 0 node corresponds to a leaf node, i.e. a subproblem GEN($\mathcal{H}, V$) in which $|\mathcal{H}| \leq 1$. Type 1 node corresponds to a subproblem of the form GEN($\mathcal{H}_{\bar{L}\setminus v}, \bar{L} \setminus v$) for some $L \subseteq V$ and $v \in \bar{L}$. Type 2 node corresponds to a subproblem of the form GEN($\mathcal{H}_{\bar{L}}, \bar{L}$) for some $L \subseteq V$. Finally, Type 3 node corresponds to a subproblem of the form GEN($\mathcal{H}_{\bar{Y}}, \bar{Y}$), for some $L \subseteq V$ and $Y \in \mathcal{H}^d_{\bar{L}}$.

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 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 most one of type 3. Associated with each active node u of $\mathbf{T}'$ is a *process* $P(u)$, which is created when u is visited for the first time, and is responsible for executing of the corresponding subproblem GEN($\mathcal{H}, V, k$) 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 happen:

(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 no 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 then the node itself then the left child. When a node u of $\mathbf{T}'$ becomes current, execution of the corresponding process $P(u)$ proceeds, form the last point it is 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 computations to return one transversal at a time. This can be done as follows.

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

- Type 3 node: suppose that u corresponds to a subproblem GEN($\mathcal{H}_{\bar{Y}}, \bar{Y}, k$), for some $L \subseteq V$ and some $Y \subseteq \mathcal{H}^d_{\bar{L}}$. When u is ready to report X, we must check first that $X \cup Y$ is a minimal transversal to $\mathcal{H}$. We have also 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 report $X \cup Y$ only if Y is the *lexicographically last* subset of $X \cup Y$ which is in $\mathcal{H}^d_{\bar{L}}$ (such a subset can be found in polynomial time).

In the following implementation, we set ϵ_1 and ϵ_2 as before, but we use a value of $k \leq 2K$ which we obtain as explained in the beginning of this section. This way we obtain an *output* quasi-polynomial time algorithm, which can be converted to an incremental one using standard techniques.

Proposition 7 ([6]) *Let $\mathcal{H} \subseteq 2^V$ be a hypergraph. Suppose that there is an algorithm P that generates all minimal transversals of $\mathcal{H}$ in time $p(|V|, |\mathcal{H}^d|)$ and space $q(|V|, |\mathcal{H}|)$, for some (quasi-)polynomials $p(\cdot,\cdot)$ and $q(\cdot)$. Then for any integer k, we can generate at least k minimal transversals of $\mathcal{H}$ (or all, if $|\mathcal{H}^d| \leq k$) in time $2|V|(p(|V|, k) + 1)$ and space $q(|V|, |\mathcal{H}|)$.*

Procedure GEN-NEXT($\mathcal{H}, V$):

1. **if** $|\mathcal{H}| = 0$, **then** return $\{\emptyset\}$;
   if $\mathcal{H} = \{H\}$, **then return** "next" element of $\{\{i\} \ : \ i \in H\}$.
2. L:= $L(\mathcal{H}, \epsilon_1)$; $\bar{L} = \{v_1, \ldots, v_l\}$.
3. **if** $|\mathcal{H}(L)| \leq \epsilon_2|\mathcal{H}|$, **then** (c.f. Corollary 2)
   for $i = 1, \ldots, l$
   W:= GEN-NEXT($\mathcal{H}_{\bar{L}\setminus v_i}, \bar{L} \setminus v_i$); Y:= $W \cup \{v_i\}$
   if $Y \in \mathcal{H}^d_{\bar{L}}$ and $Y \not\ni v_1, v_2, \ldots, v_{i-1}$, **then**
   X:= GEN-NEXT($\mathcal{H}_{\bar{Y}}, \bar{Y}$);
   compute the lexicographically last set $Z \subseteq X \cup Y$ s.t. $Z \in \mathcal{H}^d_{\bar{L}}$.
   if $Z = Y$ and $X \cup Y \in \mathcal{H}^d$, **then return** $X \cup Y$;
4. **otherwise** ($|\mathcal{H}(L)| > \epsilon_2|\mathcal{H}|$),
   L':= a balanced subset of L;
   Y:= GEN-NEXT($\mathcal{H}_{\bar{L}'}, \bar{L}'$);
   X:= GEN-NEXT($\mathcal{H}_{\bar{Y}}, \bar{Y}$);
   compute the lexicographically last set $Z \subseteq X \cup Y$ s.t. $Z \in \mathcal{H}^d_{\bar{L}}$;
   if $Z = Y$ and $X \cup Y \in \mathcal{H}^d$, **then return** $X \cup Y$.

Proposition 8 *When run to termination, Procedure GEN-NEXT($\mathcal{H}, V$) outputs all minimal transversals of $\mathcal{H}$ without repetition.*

Proof. The proof is by induction on the size of $\mathcal{H}$. If $|\mathcal{H}| \leq 1$, the statement is trivial. Assume $|\mathcal{H}| > 1$. Clearly, when the procedure outputs a minimal transversal T then $T \in \mathcal{H}^d$. Now, consider any $T \in \mathcal{H}^d$, and let L be the set computed in Step 2 of the procedure, and $\bar{L} = \{v_1, \ldots, v_l\}$. If $|\mathcal{H}(L)| \leq \epsilon_2|\mathcal{H}|$, then by Corollary 2, there exist $v = v_i \in \bar{L}$, $W \in \mathcal{H}^d_{\bar{L}\setminus v}$, $Y \in \mathcal{H}^d_{\bar{L}}$ and $X \in \mathcal{H}^d_{\bar{Y}}$, such that $Y = W \cup \{v\}$ and $T = X \cup Y$. Among all possible such i

and Y, let i be the smallest such index and let Y be the lexicographically last such subset of T. Then $Y \not\ni v_j$ for any $j \leq i-1$, otherwise, $Y \setminus v_j \in \mathcal{H}^d_{\bar{L}\setminus v_j}$, in contradiction to the minimality of i. By induction, the sets W (and hence Y) and X get returned by previous procedure calls, and thus T gets output by the procedure in Step 3. A similar argument shows that T also gets output by Step 4 of the procedure in case $|\mathcal{H}(L)| > \epsilon_2|\mathcal{H}|$.

It remains to show that the procedure does not output the same transversal more than once. Assume the opposite. Then identical transversals $T, T' \in \mathcal{H}^d$ get output by, say, Step 3 of the procedure. There exist $v_i, v_{i'} \in \bar{L}$, $W \in \mathcal{H}^d_{\bar{L}\setminus v_i}$, $W' \in \mathcal{H}^d_{\bar{L}\setminus v_{i'}}$, $Y, Y' \in \mathcal{H}^d_{\bar{L}}$, $X \in \mathcal{H}^d_{\bar{Y}}$ and $X' \in \mathcal{H}^d_{\bar{Y}'}$, such that $Y = W \cup \{v_i\}$, $Y = W' \cup \{v_{i'}\}$, $T = X \cup Y$ and $T' = X' \cup Y'$. Since the procedure outputs T only if Y is the lexicographically last subset of T which is in $\mathcal{H}^d_{\bar{L}}$, it follows that $Y = Y'$, and hence $X = X'$. By induction, for the same Y, the call X:= GEN-NEXT$(\mathcal{H}_{\bar{Y}}, \bar{Y})$ can only generate two different X's. Thus, it must be the case that Y and Y' are produced as minimal transversals of $\mathcal{H}_{\bar{L}}$ by two different calls W:= GEN-NEXT$(\mathcal{H}_{\bar{L}\setminus v_i}, \bar{L}\setminus v_i)$ and W:= GEN-NEXT$(\mathcal{H}_{\bar{L}\setminus v_{i'}}, \bar{L}\setminus v_{i'})$. If $i = i'$, such a call generates two different W', by induction. Assume therefore, without loss of generality, that $i < i'$. But then W gets generated before W', and when the procedure reaches the iteration corresponding to i', it discovers that $Y' = Y$ contains v_i and hence Y' does not get generated. □

Since processes are in one-to-one correspondence with the nodes of $\mathbf{T}$, the total number of processes generated is exactly the number of recursive calls $C(n, m, k)$, estimated in the previous section. Now we analyze the space required for the whole execution.

Analysis of memory. Clearly, we have only to bound the space used by the active subtree. Let $M(u) = M(n, m)$ be the space required by the subtree of $\mathbf{T}'$ rooted at a node u whose input $(\mathcal{H}, V)$ has size (n, m), where $n = |V|$, $m = |\mathcal{H}|$. At u we need space $O(1)$ if u has type 0, and $O(nm)$ otherwise. Thus, if u has two children p and q, then $M(u) \leq O(nm) + M(p) + M(q)$. In the worst case, p and q have respectively either types 1 and 3 (Step 3 of the procedure) or 2 and 3 (Step 4). In the former case (types 1 and 3), the input to p is $(\mathcal{H}_{\bar{L}\setminus v}, \bar{L} \setminus v)$ for some $L \subseteq V$, $v \in \bar{L}$ and the input to q is $(\mathcal{H}_{\bar{Y}}, \bar{Y})$ for $Y \in \mathcal{H}^d_{\bar{L}}$. In particular, since $\mathcal{H}_{\bar{Y}} \subseteq \mathcal{H}(L)$, the hypergraphs participating in the two subproblems are *disjoint*. Thus if $|\mathcal{H}_{\bar{L}\setminus v}| = \epsilon'|\mathcal{H}|$, then there exists a constant c such that

$$M(n, m) \leq cnm + M(n-1, \epsilon' m) + M(n-1, \epsilon'' m), \tag{22}$$

where $\epsilon' \leq 1 - \epsilon_1$ and $\epsilon'' \leq \min\{1 - \epsilon', \epsilon_2\}$. Similarly, in the latter case, we get the same recurrence as in (22) but with $1 - \epsilon_2 \leq \epsilon' \leq 1 - (\epsilon_2 - \epsilon_1)$ and $\epsilon'' \leq 1 - \epsilon'$. In both cases we get $\epsilon' \leq 1 - \epsilon$, if we set $\epsilon_2 = 2\epsilon_1 = 2\epsilon$.

We prove by induction on $m \geq 1$ that $M(n, m) \leq cnm(\log m + 1)/H(\epsilon)$, where $H(\cdot)$ is given by (4). For $m = 1$, the statement holds, assuming c is large enough such that

$M(n,1) \leq c$. For $m \geq 2$, we get by induction that

$$M(n,m) \leq cnm + \frac{cn\epsilon' m(\log(\epsilon' m)+1)}{H(\epsilon)} + \frac{cn\epsilon'' m(\log(\epsilon'' m)+1)}{H(\epsilon)}$$
$$\leq \frac{cnm(\log m + 1)}{H(\epsilon)} + cnm(1 - \frac{1}{H(\epsilon)}(\epsilon' \log\frac{1}{\epsilon'} + \epsilon'' \log\frac{1}{\epsilon''})). \quad (23)$$

Suppose first that $\epsilon' \in [1-2\epsilon, 1-\epsilon]$. Then by the concavity of $H(x)$ for $x \in [0,1]$ we know that $H(\epsilon') \geq \min\{H(1-2\epsilon), H(1-\epsilon)\}$. Since $\epsilon \leq 1/3$ (c.f. (2) and (3)), we also have $H(1-2\epsilon) \geq H(1-\epsilon)$, and we conclude that $H(\epsilon') \geq H(1-\epsilon) = H(\epsilon)$. Using $\epsilon'' \leq 1-\epsilon'$ in (23) we get the claimed bound in this case. Similarly, if $\epsilon' < 1-2\epsilon$, then using $\epsilon'' \leq \epsilon_2 = 2\epsilon$, we get again the claimed bound from (23) since $H(2\epsilon) \geq H(\epsilon)$. □

References

[1] M. Akra and L. Bazzi, On the solution of linear recurrence equations, *Comput. Optim. Appl.* 10 (1998), no. 2, 195–210.

[2] M. Anthony and N. Biggs, *Computational Learning Theory*, Cambridge Univ. Press, 1992.

[3] C. Berge, *Hypergraphs*, North Holland Mathematical Library, Vol. 445, 1989.

[4] J. C. Bioch and T. Ibaraki, Complexity of identification and dualization of positive Boolean functions, *Information and Computation* 123 (1995) 50-63.

[5] E. Boros, K. Elbassioni, V. Gurvich, and L. Khachiyan, An efficient incremental algorithm for generating all maximal independent sets in hypergraphs of bounded dimension, *Parallel Processing Letters*, **10** (2000), pp. 253–266.

[6] E. Boros, K. Elbassioni, V. Gurvich and L. Khachiyan, Generating Maximal Independent Sets for Hypergraphs with Bounded Edge-Intersections, in *Proc. the 6th Latin American Theoretical Informatics Conference (LATIN 2004)*, *LNCS* 2976, pp. 488–498.

[7] 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. Comput.*, **31** (5) (2002) pp. 1624–1643.

[8] E. Boros, K. Elbassioni, V. Gurvich and L. Khachiyan, Matroid Intersections, Polymatroid Inequalities, and Related Problems. In *the Proceedings of the 27th International Symposium on Mathematical Foundations of Computer Science (MFCS 2002)*, *LNCS* 2420, pp. 143–154, Poland, August 2002.

[9] E. Boros, K. Elbassioni, V. Gurvich and L. Khachiyan, An inequality for polymatroid functions and its applications, *Discrete Applied Mathematics* 131(2): 255-281 (2003).

[10] E. Boros, V. Gurvich, and P.L. Hammer, Dual subimplicants of positive Boolean functions, *Optimization Methods and Software*, 10 (1998) 147-156.

[11] P. Bertolazzi and A. Sassano, An $O(mn)$ time algorithm for regular set-covering problems, *Theoretical Computer Science* 54 (1987) 237-247.

[12] C. J. Colbourn, *The combinatorics of network reliability*, Oxford Univ. Press, 1987.

[13] Y. Crama, Dualization of regular Boolean functions, *Discrete Applied Math.* 16 (1987) 79-85.

[14] E. Dahlhaus and M. Karpinski, A fast parallel algorithm for computing all maximal cliques in a graph and the related problems, *Proc. 1st Scandinavian Workshop on Algorithm Theory (SWAT)* , Sweden, July 5-8, 1988, LNCS 318, pp. 139-144.

[15] C. Domingo, Polynomial time algorithms for some self-duality problems, in *Proceedings of the 3rd Italian Conference on Algorithms and Complexity (CIAC'97)*, Rome, Italy, March 12-14, 1997, pp. 171-180.

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

[17] T. Eiter, Exact Transversal Hypergraphs and Application to Boolean μ-Functions, *J. Symb. Comput.* 17(3): pp. 215–225 (1994).

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

[19] T. Eiter, G. Gottlob and K. Makino, New results on monotone dualization and generating hypergraph transversals, in *the 34th Annual ACM Symp. Theory of Computing (STOC)*, 2002.

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

[21] D. Eppstein, All maximal independent sets and dynamic dominance for sparse graphs, *SODA 2005*, pp. 451-459.

[22] M. L. Fredman and L. Khachiyan, On the complexity of dualization of monotone disjunctive normal forms. *J. Algorithms*, 21 (1996) 618-628.

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

[24] D. Gaur and R. Krishnamurti, Self-duality of bounded monotone Boolean functions and related problems, in *Proceedings of the 11th International Conference on Algorithmic Learning Theory (ALT)*, Lecture Notes in Comput. Sci. 1968, Springer-Verlag, New York, 2000, pp. 209–223.

[25] G. Gogic, C. Papadimitriou, and M. Sideri, Incremental recompilation of knowledge, *J. Artificial Intelligence Res.*, 8 (1998), pp. 23–37.

[26] G. Gottlob, Hypergraph transversals, *Proceedings of the 3rd International Symposium of Foundations of Information and Knowledge Systems (FoIKS 2004)*, Wilhelminenburg Castle, Austria, February 17-20, 2004, Lecture Notes in Computer Science 2942, pp. 1-5.

[27] D. Gunopulos, R. Khardon, H. Mannila, and H. Toivonen, Data mining, hypergraph transversals and machine learning, in *Proc. the 16th ACM-SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS 1997)*, pp. 12–15.

[28] V. Gurvich, Nash-solvability of games in pure strategies, *USSR Comput. Math and Math. Phys.* **15-2** (1975) 357-371.

[29] V. Gurvich and L. Khachiyan, On generating the irredundant conjunctive and disjunctive normal forms of monotone Boolean functions, *Discrete Applied Mathematics*, 1996-97, issue 1-3, (1999) 363-373.

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

[31] D. S. Johnson, M. Yannakakis and C. H. Papadimitriou, On generating all maximal independent sets, *Info. Process. Lett.*, 27 (1988) 119-123.

[32] R. Karp and V. Ramachandran, Parallel algorithms for shared memory machines, in Handbook of Theoretical Computer Science, J. van Leeuwen, ed., North Holland (1990) 869-941.

[33] R. Karp, E. Upfal, and A. Wigderson, The complexity of parallel search, *Journal of Computer and System Science*, 36 (1988) 225-253.

[34] D. J. Kavvadias, C. Papadimitriou, and M. Sideri, On Horn envelopes and hypergraph transversals, in *Proceedings of the 4th International Symposium on Algorithms and Computation (ISAAC-93)*, Lecture Notes in Comput. Sci. 762, W. Ng, ed., Springer-Verlag, New York, 1993, pp. 399–405.

[35] D. J. Kavvadias and E. C. Stavropoulos, The nondeterministic complexity of the Fredman and Khachiyan algorithm for the dualization of monotone DNFs, Technical Report CTI TR 2001.12.03, Computer Technology Institute, Patras, Greece, December 2001.

[36] C. M. R. Kintala and P. C. Fischer, Refining nondeterminism in relativized polynomial-time bounded computations, *SIAM J. Comput.*, 9 (1980), pp. 46–53.

[37] L. Khachiyan (2000), Transversal hypergraphs and families of polyhedral cones. In: N. Hadjisavvas and P. Pardalos (Eds.), *Advances in Convex Analysis and Global Optimization, Honoring the memory of K. Carathéodory*, pp. 105–118, Kluwer Academic Publishers, Dordrecht/Boston/London.

[38] L. Khachiyan, E. Boros, K. Elbassioni, V. Gurvich, and K. Makino, On the Complexity of Some Enumeration Problems for Matroids, to appear in SIAM Journal on Discrete Mathematics.

[39] L. Khachiyan, E. Boros, K. Elbassioni, V. Gurvich, A New Algorithm for the Hypergraph Transversal Problem, in *the Proceedings of the Computing and Combinatorics, 11th Annual International Conference (COCOON 2005)*, Kunming, China, August 16-29, 2005, Lecture Notes in Computer Science 3595, pp. 767-776.

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

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

[42] K. Makino, Efficient Dualization of $O(\log n)$-Term Monotone Disjunctive Normal Forms, *Discrete Applied Mathematics*, 126 (2003) pp. 305–312.

[43] K. Makino and T. Ibaraki, A fast and simple algorithm for identifying 2-monotonic positive Boolean functions, *Journal of Algorithms*, 26 (1998) 291-305.

[44] H. Mannila and K. J. Räihä, Design by example: An application of Armstrong relations, *Journal of Computer and System Science* 22 (1986) 126-141.

[45] C. Papadimitriou, NP-completeness: A retrospective, in Proceedings of the 24th International Colloquium on Automata, Languages, and Programming (ICALP '97), Lecture Notes in Comput. Sci. 1256, Springer-Verlag, New York, 1997, pp. 2–6.

[46] U. N. Peled and B. Simeone, An $O(nm)$-time algorithm for computing the dual of a regular Boolean function, *Discrete Applied Mathematics* 49 (1994) 309-323.

[47] N. M. L. Pitt, Generating all maximal independent sets of bounded-degree hypergraphs, in Proceedings of the 10th Annual Conference on Computational Learning Theory (COLT), Nashville, TN, 1997, pp. 211–217.

[48] K. G. Ramamurthy, *Coherent Structures and Simple Games*, Kluwer Academic Publishers, 1990.

[49] R. C. Read, Every one a winner, or how to avoid isomorphism when cataloging combinatorial configurations, *Annals of Discrete Mathematics* 2 (1978) 107-120.

[50] K. Takata, On the sequential method for listing minimal hitting sets, in *Proc. SIAM Workshop on Discrete Mathematics and Data Mining (DM & DM)*, Arlington, VA, April 2002, pp. 109–120.

[51] H. Tamaki, Space-efficient enumeration of minimal transversals of a hypergraph, IPSJ-AL **75** (2000) 29–36.

[52] V. D. Thi, Minimal keys and antikeys, *Acta Cybernet.*, 7 (1986), pp. 361–371.