I N F S Y S

R E S E A R C H

R E P O R T

INSTITUT FÜR INFORMATIONSSYSTEME

ARBEITSBEREICH WISSENSBASIERTE SYSTEME

COMPUTATIONAL ASPECTS OF MONOTONE DUALIZATION: A BRIEF SURVEY

Thomas Eiter Kazuhisa Makino Georg Gottlob

INFSYS RESEARCH REPORT 1843-06-01

JANUARY 2006

Institut für Informationssysteme
AB Wissensbasierte Systeme
Technische Universität Wien
Favoritenstraße 9-11
A-1040 Wien, Austria
Tel: +43-1-58801-18405
Fax: +43-1-58801-18493
sek@kr.tuwien.ac.at
www.kr.tuwien.ac.at

INFSYS RESEARCH REPORT

INFSYS RESEARCH REPORT 1843-06-01, JANUARY 2006

COMPUTATIONAL ASPECTS OF MONOTONE DUALIZATION: A BRIEF SURVEY

Thomas Eiter[1], Kazuhisa Makino[2], and Georg Gottlob[3]

Abstract. Dualization of a monotone Boolean function represented by a conjunctive normal form (CNF) is a problem which, in different disguise, is ubiquitous in many areas including Computer Science, Artificial Intelligence, and Game Theory to mention some of them. It is also one of the few problems whose precise tractability status (in terms of polynomial-time solvability) is still unknown, and now open for more than 25 years. In this paper, we briefly survey computational results for this problem, where we focus on the famous paper by Fredman and Khachiyan (J. Algorithms, 21:618–628, 1996), which showed that the problem is solvable in quasi-polynomial time (and thus most likely not co-NP-hard), as well as on follow-up works. We consider computational aspects including limited nondeterminism, probabilistic computation, parallel and learning-based algorithms, and implementations and experimental results from the literature. The paper closes with open issues for further research.

Keywords: Dualization, Monotone Boolean Functions, Hypergraphs, Transversals, Hitting Sets, Independent Sets, Set Coverings, Self-duality, Output-polynomial Algorithms, Polynomial-total Time, Quasi-polynomial Time, Combinatorial Enumeration, Limited Nondeterminism.

---

[1]Institute of Information Systems, Knowledge-Based Systems Group, TU Vienna, Favoritenstraße 9-11, A-1040 Vienna, Austria. E-mail: eiter@kr.tuwien.ac.at.

[2]Department of Mathematical Informatics, Graduate School of Information and Technology, University of Tokyo, Tokyo, 113-8656, Japan. E-mail: makino@mist.i.u-tokyo.ac.jp.

[3]Computing Laboratory, Oxford University, Parks Road, Oxford, OX1 3QD, United Kingdom. E-mail: Georg.Gottlob@comlab.ox.ac.uk.

Acknowledgements: This work has been partly supported by visit grants of TU Wien and the Wolfgang Pauli Institute, Vienna, and by a Grant-in-Aid of the Ministry of Education, Science, Sports and Culture of Japan.

Contents

1 Introduction

Dualizing a Boolean function $f = f(x_1, x_2, \ldots, x_n)$ is a well-known problem in discrete mathematics. A Boolean formula for the dual function $f^d = \overline{f}(\overline{x}_1, \overline{x}_2, \ldots, \overline{x}_n)$ can be easily obtained by interchanging in any Boolean formula φ representing f the connectives $\wedge$ and $\vee$, as well as 0 and 1. However, an efficient solution of this problem is not straightforward if the formulas must be in a special format such as conjunctive normal form (CNF) $\varphi = \bigwedge_{c \in C} \bigvee_{\ell_j \in c}$, where each ℓ_j is a variable or its negation, which in addition may be requested to be prime, i.e., no literal ℓ_j can be removed from each clause c. For example, the dual of the Boolean function $f = x_2(x_1 \vee x_3)(x_1 \vee x_4)$ is $f^d = x_2 \vee x_1x_3 \vee x_1x_4$, which has the prime CNF $(x_2 \vee x_1)(x_2 \vee x_3 \vee x_4)$.

Constructing a dual CNF by applying elementary Boolean laws is not efficient in general. The study of advanced dualization methods (possibly for restricted classes of Boolean functions) dates back to at least the 1950/60's, cf. [61, 4, 43]. Among these classes, the monotone Boolean functions f, which satisfy that $f(x) \leq f(y)$ whenever $x \leq y$, have received a lot of attention. Monotone functions have many nice properties, including that they have a unique prime CNF in which no negation occurs. Following [31], the monotone dualization problem is formulated as follows:[1]

Problem DUALIZATION

Input: The prime CNF φ of a monotone Boolean function $f = f(x_1, \ldots, x_m)$.
Output: The prime CNF ψ of its dual $f^d = \overline{f}(\overline{x}_1, \ldots, \overline{x}_m)$.

This problem is, in different disguise, ubiquitous and many problems in Computer Science, Artificial Intelligence, and Game Theory are easily reduced to it, cf. [7, 29, 30, 32, 46]. Since $(f^d)^d = f$, it is tantamount to generating a prime disjunctive normal form (DNF) for f from a given CNF. Because this DNF is unique and consists of all prime implicants of f, monotone dualization resorts to an instance of the classical problem of generating the prime implicants of a Boolean function, for which numerous algorithms exist, cf. [54, 76, 73] and references therein.

Furthermore, monotone dualization is intimately related to important problems in hypergraph theory. It is well-known that DUALIZATION is equivalent to computing the *transversal hypergraph* $Tr(\mathcal{H})$ [5] of a (finite) hypergraph $\mathcal{H} = (V, \mathcal{E})$, which has the same vertices as $\mathcal{H}$ and as edges the minimal *transversals* (or *hitting sets*) of $\mathcal{H}$, i.e., the minimal (under $\subseteq$) sets $T \subseteq V$ such that $T \cap E \neq \emptyset$, for each $E \in \mathcal{E}$. For that, the variables are identified with the vertices V and the edges of $\mathcal{H}$ with the sets of literals in the clauses of φ. E.g., for the function f above, $V = \{x_1, x_2, x_3, x_4\}$ and $\mathcal{E} = \{\{x_2\}, \{x_1, x_3\}, \{x_1, x_4\}\}$; the minimal transversals are $\{x_1, x_2\}$ and $\{x_2, x_3, x_4\}$, which correspond to the dual prime CNF $(x_1 \vee x_2)(x_2 \vee x_3 \vee x_4)$. Since the complements of the transversals of $\mathcal{H}$ are its *independent sets*, DUALIZATION is tantamount to computing the maximal independent sets of a hypergraph (in our example, $\{x_3, x_4\}$ and $\{x_1\}$) [57, 49]. Furthermore, each transversal of $\mathcal{H}$ corresponds to a *set covering* [58] of the dual hypergraph $\mathcal{H}^d = (V^d, \mathcal{E}^d)$ of $\mathcal{H}$, with vertices $V^d = \mathcal{E}$ and edges $\mathcal{E}^d = \{E_v \mid v \in V\}$, where $E_v = \{E \in \mathcal{E} \mid v \in E\}$ is the set of edges of $\mathcal{H}$ in which v occurs, for each $v \in V$. In our example, $\{x_1, x_2\}$ and $\{x_2, x_3, x_4\}$ correspond to the set coverings $\{E_{x_2}, E_{x_1}\}$ and $\{E_{x_2}, E_{x_3}, E_{x_4}\}$, respectively (recall that $\mathcal{C} \subseteq \mathcal{E}^d$ is a set covering of $\mathcal{H}^d$, if

[1] As for tractability concerns, the input may in fact consist of any CNF for f, from which its unique prime CNF is easily computed.

$\bigcup \mathcal{C} = V^d$). Thus, DUALIZATION is tantamount to computing all minimal set coverings of $\mathcal{H}^d$, which can be efficiently computed from $\mathcal{H}$, (vertices x_i and x_j such that $E_{x_i} = E_{x_j}$ can be factored out easily).

In the 1980's and early 1990's, monotone dualization and its many relatives have been intensively studied with respect to their intrinsic computational complexity, and in particular the question whether these problems are tractable. To this end, the monotone dualization problem has been cast, as common in complexity theory, to a decisional version:

Problem DUAL

Input: Prime CNFs φ, ψ of monotone Boolean functions $f = f(x_1, \ldots, x_m)$ and $g = g(x_1, \ldots, x_m)$, respectively.
Question: Are f and g dual Boolean functions?

Problem DUALIZATION is (under a suitable notion) polynomial-time equivalent to problem DUAL [7, 46]. While DUAL is easily seen to belong to the class co-NP, it is open (for more than 25 years now, cf. [57, 48, 7, 29, 68]), whether the problem is solvable in polynomial time. Many polynomial cases are known, cf. [2, 6, 8, 14, 12, 20, 9, 21, 26, 27, 28, 33, 31, 62, 63, 64, 67, 69, 70] and references therein, but also interesting cases which are as hard as the general problem. The most noticeable is deciding whether $f^d = f$, which is known as SELF-DUALITY. A canonical example of a self-dual function is $f = (x_1 \vee x_2)(x_2 \vee x_3)(x_1 \vee x_3)$. In terms of hypergraphs, self-duality amounts to $Tr(\mathcal{H}) = \mathcal{H}$ [74] (see also [7, 29]), or whether an intersecting $\mathcal{H}$ is *edge-critical*, i.e., not 2-colorable but 2-colorable if any edge is removed (such hypergraphs are called strange hypergraphs in [60]; see [25, 39] for polynomial cases of self-duality, and [29] for other classes of hypergraphs for which 2-colorability is polynomially equivalent to DUAL).

In 1996, Fredman and Khachiyan proved in the landmark paper [37] that problem DUAL is solvable in quasi-polynomial time, i.e., in time $O(n^{\mathrm{polylog}(n)})$, where n is the size of the input. They considered two algorithms called A and B (which we shall briefly recall in Section 4), and demonstrated in a clever analysis that their running time is bounded by $n^{O(\log^2 n)}$ and $n^{o(\log n)}$, respectively. This result provides strong evidence that problem DUAL is not co-NP-hard, since it is widely believed that no co-NP-hard problem can be solved within this time bound. Furthermore, it gives rise to algorithms which solve problem DUALIZATION in *quasi-polynomial total time* [37, 46], i.e., in quasi-polynomial time in the combined size of φ and ψ. Note that since the output ψ can be exponentially larger than the input φ (for an easy example, let $\varphi = (x_1 \vee x_2)(x_3 \vee x_4) \cdots (x_{2n-1} \vee x_{2n})$), tractability of DUALIZATION should be assessed in terms of solvability in *polynomial total* (or *output-polynomial* [49]) *time*, i.e., in polynomial time in the combined size of φ and ψ. The known polynomial cases of DUAL give rise to the respective output-polynomial cases of DUALIZATION.

The results of Fredman and Khachiyan have been very influential, not only because of their significance with respect to the issue of tractability of DUAL, but also because of the combinatorial methods and tools used in it. Several follow-up works have been based on these results, addressing different aspects of the problem. Among them are tractability under different input representations [46], solvability in polynomial time with limited nondeterminism [52, 31], probabilistic aspects such as polynomial-time solvability on average [75, 40], generalization of the problems to integer linear systems [15], etc.

In this paper, we briefly survey computational aspects of monotone dualization. The survey is not exhaustive and focuses on the Fredman-Khachiyan results and follow-up work on dualization which has

been strongly influenced by them. Other algorithms, in particular ones based on different computation paradigms, are treated more shortly; for applications of dualization and closely related problems in different domains, we refer to [7, 29, 30, 32, 37, 46] and references therein.

The plan for the remainder of this paper is as follows. The next section recalls concepts and fixes notation. In Section 3 we then recall some simple decomposition-based algorithms which have been refined and improved in many work in the literature. Section 4 is devoted to the famous Fredman and Khachiyan paper [37], after which we briefly discuss some of the many follow-up works in Section 5. In Section 6 we consider other algorithms, where for brevity we focus in learning-based, ordered variable-decomposition based, and parallel algorithms. Section 7 gathers some papers with experimental results of the algorithms and variants, while Section 8 concludes the paper with a discussion and open issues.

2 Preliminaries

Recall that a *Boolean function* (in short, *function*) is a mapping $f : \{0,1\}^n \to \{0,1\}$, where $v \in \{0,1\}^n$ is called a *Boolean vector* (in short, *vector*) whose i-th component is denoted by v_i. As usual, we write $g \leq f$ if f and g satisfy $g(v) \leq f(v)$ for all $v \in \{0,1\}^n$, and $g < f$ if $g \leq f$ and $g \neq f$. A function f is *monotone* (or *positive*), if $v \leq w$ (i.e., $v_i \leq w_i$ for all i) implies $f(v) \leq f(w)$ for all $v, w \in \{0,1\}^n$. Boolean variables $x_1, x_2, \ldots, x_n$ and their complements $\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n$ are called *literals*. A *clause* (resp., *term*) is a disjunction (resp., conjunction) of literals containing at most one of x_i and $\bar{x}_i$ for each variable. The empty disjunction (resp., conjunction) is denoted by $\bot$ (resp. $\top$).

A clause c (resp., term t) is an *implicate* (resp., *implicant*) of a function f, if $f \leq c$ (resp., $t \leq f$); moreover, it is *prime*, if there is no implicate $c' < c$ (resp., no implicant $t' > t$) of f, and *monotone*, if it consists of positive literals only. A *conjunctive normal form* (CNF) (resp., disjunctive normal form, DNF) is a conjunction of clauses (resp., disjunction of terms); it is *prime* (resp. *monotone*), if all its members are prime (resp. *monotone*). For any CNF (resp., DNF) ρ, we denote by $|\rho|$ the number of clauses (resp., terms) in it. Furthermore, for any formula φ, we denote by $V(\varphi)$ the set of variables that occur in φ, and by $\|\varphi\|$ its *length*, i.e., the number of literals in it. It is convenient to view CNFs φ also as sets of clauses, and clauses as sets of literals; we thus use respective notation (e.g., $c \in \varphi$, $\overline{x}_1 \in c$ etc).

As well-known, a function f is monotone iff it has a monotone CNF. Furthermore, all prime implicants and prime implicates of a monotone f are monotone, and it has a unique prime CNF, denoted $\mathrm{CNF}(f)$, which is given by the conjunction of all its prime implicates. For example, the monotone f such that $f(v) = 1$ iff $v \in \{(1100), (1110), (1101), (0111), (1111)\}$ has $\mathrm{CNF}(f) = x_2(x_1 \vee x_3)(x_1 \vee x_4)$ (see Figure 1).

Recall that the *dual* of a function f, denoted f^d, is defined by $f^d(x) = \overline{f}(\overline{x})$, where $\overline{f}$ and $\overline{x}$ is the complement of f and x, respectively. By definition, we have $(f^d)^d = f$. From De Morgan's law, we obtain from formula φ of f a formula φ^d for f^d by exchanging $\vee$ and $\wedge$ as well as the constants 0 and 1. For example, if f is given by $\varphi = x_1x_2 \vee \overline{x}_1(\overline{x}_3 \vee x_4)$, then f^d is represented by $\psi = (x_1 \vee x_2)(\overline{x}_1 \vee \overline{x}_3x_4)$. Note that for any monotone function f, f^d is monotone as well. Thus if $\mathrm{CNF}(f^d) = \bigwedge_{c \in C}(\bigvee_{x_i \in c} x_i)$, then f has, by De Morgan's law, the unique prime DNF $\bigvee_{c \in C}(\bigwedge_{x_i \in c} x_i)$, which we denote by $\mathrm{DNF}(f)$. E.g., f from Figure 1 has $\mathrm{DNF}(f) = x_1x_2 \vee x_2x_3x_4$. Thus, we will regard DUALIZATION also as the problem of computing $\mathrm{DNF}(f)$ from $\mathrm{CNF}(f)$. Note that since $(f^d)^d = f$, computing $\mathrm{CNF}(f)$ from the $\mathrm{DNF}(f)$ (resp., duality testing for DNFs in the input) has same complexity.

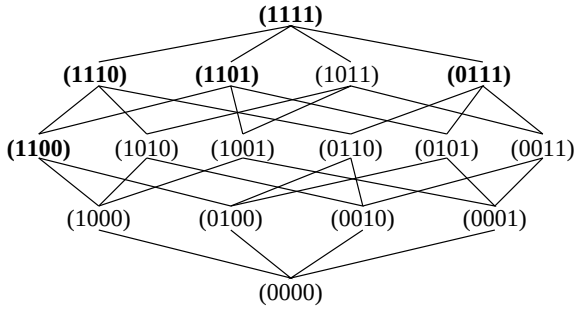

Figure 1: Boolean lattice and function $f = x_2(x_1 \vee x_3)(x_1 \vee x_4)$ (vectors v such that $f(v) = 1$ in bold)

3 Simple Algorithms

The dual form may be computed by simple divide and conquer algorithms using different decomposition methods, such as clause-based and variable-based decomposition.

Clause-based decomposition. This method aims at combining the dual forms of sub-CNFs of the input CNF φ to obtain the result. A simple example is the algorithm referred to as *Berge's algorithm* [5], which works as follows ($\varphi = \bigwedge_{i=1}^{m} c_i$):

1. Initialize $\psi_0 := \{\emptyset\} \quad (= \bot)$;
2. for each $i = 1, 2, \ldots, m$ do $\psi_i := \min(\{c' \cup \{x\} \mid c' \in \psi_{i-1}, x \in c_i\})$, where $\min(\mathcal{S})$ is the set of all minimal sets (wrt $\subseteq$) in the family of sets $\mathcal{S}$.

Then, $\psi = \psi_m$ holds. We remark that this algorithm has appeared in the literature earlier e.g. in [61, 72, 58], and several improvements have been proposed, e.g. [4, 51, 3, 82]; see also [27]. For the input $\varphi = x_2(x_1 \vee x_3)(x_1 \vee x_4)$, we obtain (for left-to-right edge ordering) $\psi_0 = \bot$, $\psi_1 = x_2$, $\psi_2 = (x_1 \vee x_2)(x_2 \vee x_3)$, and $\psi_3 = (x_1 \vee x_2)(x_2 \vee x_3 \vee x_4) = \psi$. There are simple examples which show that the Berge algorithm may produce intermediate CNFs ψ_i of exponential size in the final ψ. E.g., let φ represent the complete graph on n vertices $x_1, \ldots, x_n$, and suppose that the clauses (i.e., edges) $c_1, \ldots, c_m$, $m = \binom{n}{2}$, are ordered such that $c_j = x_{2j-1} \vee x_{2j}$, for $j = 1, 2, \ldots, n/2$. Then $\psi_{n/2}$ has $\Omega(2^{n/2})$ clauses (of the form $\ell_1 \vee \cdots \vee \ell_{n/2}$ where $\ell_j \in \{x_{2j-1}, x_{2j}\}$ for each $j = 1, 2, \ldots, n/2$), but $\psi_m = \psi$ has only n clauses, which are of the form $c'_i = \bigvee_{j \in I_i} x_j$, where $I_i = \{1, \ldots, n\} \backslash \{i\}$, for $i = 1, 2, \ldots, n$. In this example, using a suitable ordering the algorithm works polynomially. However, as shown by Takata [78], there are instances φ for which the largest intermediate result has size $t^{\Omega(\log \log t)}$, where t is the size of φ and ψ, for every possible edge-ordering. Hence, the Berge algorithm has super-polynomial runtime in general.

Variable-based decomposition. Different from edge-based decomposition, this method aims at combining the dual forms of CNFs which do or do not contain a particular variable x. This can be done using the Shannon decomposition of f, given by $f = x f_{x=1} \vee f_{x=0}$. It implies that

$$\varphi^d \equiv (x \vee {\varphi_{\not\ni x}}^d) \wedge {\varphi_{-x}}^d, \tag{1}$$

where $\varphi_{\not\ni x} = \{c \in \varphi : x \notin c\}$ are the clauses in φ not containing x, and $\varphi_{-x} = \{c - \{x\} \mid c \in \varphi\}$ are the clauses in φ minus the variable x. We can thus compute φ as follows:

1. If $\varphi = \top$ then return $\psi := \bot$.
2. If $\varphi = \bot$ then return $\psi := \top$.
3. Otherwise, select a variable x, and recursively compute $\psi_1 := {\varphi_{\not\ni x}}^d$ and $\psi_2 := {\varphi_{-x}}^d$.
4. Return $\psi := \min(\{x \vee c \mid c \in \psi_1\} \cup \psi_2)$.

For the input $\varphi = x_2(x_1 \vee x_3)(x_1 \vee x_4)$ and decomposition by x_1, we have $\psi_1 = x_2$ and $\psi_2 = x_2x_3x_4$; therefore, $\psi = \min(\{x_2 \vee x_1,\, x_2 \vee x_3 \vee x_4\}) = (x_2 \vee x_1)(x_2 \vee x_3 \vee x_4)$ is recursively computed. For the CNF φ which corresponds to the complete graph on $x_1, \ldots, x_n$, decomposition by x_1 yields for $\varphi_{\not\ni x_1}$ a recursive instance, viz. the complete graph on $x_2, \ldots, x_n$, and for φ_{-x_1} a formula such that $\min(\varphi_{-x_1})$ = $x_2 \wedge x_3 \wedge \cdots \wedge x_n$; the formula ${\varphi_{\not\ni x_1}}^d$ consists of the clauses $c_i = \{x_j \mid j \in \{2, \ldots, n\} - \{i\}\}$, $i \in \{2, 3, \ldots, n\}$, and ${\varphi_{-x_1}}^d$ of the single clause $x_2 \vee x_3 \vee \cdots \vee x_n$. While on complete graphs, the algorithm is output-polynomial (in fact, polynomial in the input size), one can easily construct instances which disprove that it is output-polynomial in general.

Also for this decomposition method, many refinements and variants exist, e.g. [3, 31, 27, 66]. Another decomposition scheme, which derives from (1), is

$$\varphi^d \equiv \bigwedge_{i=1}^{n} (x_i \vee ((\varphi_{\not\ni x_i})_{-x_1,x_2,\ldots,x_{i-1}})^d), \tag{2}$$

where $\alpha_{-x_1,x_2,\ldots,x_k} = \{c - \{x_1, \ldots, x_k\} \mid c \in \alpha\}$, according to which the dual clauses are partitioned into those containing x_1; those containing x_2 but not x_1; those containing x_3 but not x_2, x_2 etc. This scheme is called "sprouting rule" in the context of generating prime implicants [76], and has been used e.g. in [3, 66, 73].

4 Fredman and Khachiyan's Results

In their paper [37], Fredman and Khachiyan have advanced variable-based decomposition for dualization testing, and have presented two algorithms, A and B, which solve the problem with decreasing complexity.

Algorithm A. the first algorithm, which is shown in Table 1 (reformulated for CNF input), has running time bounded by

$$n^{4n^2+O(1)} \;=\; n^{O(\log^2 n)}, \tag{3}$$

where n is the size of the input. It is based on two important enhancements of variable-based decomposition:

1. Exploit simple properties of monotone dual pairs (intersection properties, relations of clause numbers and sizes, etc), by which simple instances of the problem can be resolved.

2. Select for the decomposition a benign variable which ensures that an exponential explosion blowup in terms of (recursive) subproblems will not happen. As Fredman and Khachiyan showed, "frequent" variables (those which occur most often in the input) fulfill this role.

In order to obtain analytic result (3) for the algorithm, the following properties of dual pairs are critical. Let φ and ψ be CNFs representing functions f and f^d, respectively. Then we have

$$\Sigma_{c\in\varphi}\, 2^{-|c|} + \Sigma_{c'\in\psi}\, 2^{-|c'|} \geq 1.$$

We note that this property holds for arbitrary functions, not necessarily monotone functions. This immediately implies that φ or ψ contains a short clause, i.e.,

$$\min\{|c| \mid c \in \varphi \cup \psi\} \leq \log(|\varphi| + |\psi|). \tag{4}$$

The second property is the intersection property of dual pairs:

$$c \cap c' \neq \emptyset \text{ for any pair of } c \in \varphi \text{ and } c' \in \psi. \tag{5}$$

From (4) and (5), there exists a variable x_i whose frequency $\epsilon(x_i) = \max\{\, |\{c \in \varphi \mid x_i \in c\}|/|\varphi|, |\{c \in \psi \mid x_i \in c\}|/|\psi| \,\}$ satisfies

$$\epsilon(x_i) \;\geq\; 1/\log(|\varphi| + |\psi|) \tag{6}$$

(the bounds (6) and (4) are tight within a factor of 2 [45]). Therefore, one of the subproblems (A.1) and (A.2) created by decomposition with x_i in Algorithm A is small, from which we can prove that Algorithm A runs in $n^{O(\log^2 n)}$ time and thus is quasi-polynomial.

However, Algorithm A is not polynomial. For the class of monotone read-once functions, i.e., Boolean functions which can be represented by an expression in which each variable occurs at most once, Gurvich and Khachiyan [46] report an $m^{\Omega(\log m)}$ lower bound for Algorithm A, where $m = |\varphi| + |\psi|$. Notice that dualization of read-once functions is solvable in polynomial time, cf. [2, 28]; thus, while quasi-polynomial in general, Algorithm A is not always optimal.

Algorithm B. The second algorithm in [37], Algorithm B, checks for duality in $n^{o(\log n)}$ time. More precisely, it works in time

$$n^{4\chi(n)+O(1)}, \tag{7}$$

where $\chi(n)^{\chi(n)} = n$; note that $\chi(n) \sim \log n/\log\log n = o(\log n)$. Here, the idea is to take in the decomposition step also information about the solvability of the one subproblem for the other into account.

Briefly, similar as Algorithm A in steps 1-3, Algorithm B first deals with instances which can be decided easily. For the remaining cases, it selects any variable x for decomposition of the problem. If x is frequent in both input formulas, then it applies the decomposition scheme of Algorithm A for x. Otherwise, it uses a different decomposition scheme by which the problem is quasi-polynomially solved. Depending on whether x is infrequent in φ or in ψ, first duality of the pair (A.1) respectively (A.2) as in Algorithm A is tested. If the result for (A.1) is positive, then instead of the single pair in (A.2) for each clause $c \in \psi_0$ the pair (ψ_1^c, φ_0^c) is tested for duality, where $\psi_1^c = \{c' \in \psi_1 \mid c \cap c' = \emptyset\}$ and $\varphi_0^c = \{c' \setminus c \mid c' \in \varphi_0\}$; similarly if (A.2) is positive.

Algorithm A.

Input: Monotone CNFs φ, ψ representing monotone f, g s.t. $V(c) \cap V(c') \neq \emptyset$, for all $c \in \varphi$, $c' \in \psi$.

Output: `yes` if $f = g^d$, otherwise a vector w of form $w = (w_1, \ldots, w_m)$ such that $f(w) \neq g^d(w)$.

Step 1:

Delete all redundant (i.e., non-minimal) clauses from φ and ψ.

Step 2:

Check whether (i) $V(\phi) = V(\psi)$,
(ii) $\max_{c \in \varphi} |c| \leq |\psi|$,
(iii) $\max_{c' \in \psi} |c'| \leq |\varphi|$, and
(iv) $\Sigma_{c \in \varphi} 2^{-|c|} + \Sigma_{c' \in \psi} 2^{-|c'|} \geq 1$.

If any of (i)-(iv) fails, $f \neq g^d$ and a witness w is found in polynomial time (cf. [37]).

Step 3:

If $|\varphi| \cdot |\psi| \leq 1$, test duality in $O(1)$ time.

Step 4:

If $|\varphi| \cdot |\psi| \geq 2$, find a variable x_i occurring in φ or ψ with frequency $\geq 1/\log(|\varphi| + |\psi|)$.
Let

$$\varphi_0 = \{c - \{x_i\} \mid x_i \in c,\, c \in \varphi\}, \quad \varphi_1 = \{c \mid x_i \notin c,\, c \in \varphi\},$$
$$\psi_0 = \{c' - \{x_i\} \mid x_i \in c',\, c' \in \psi\},\ \psi_1 = \{c' \mid x_i \notin c',\, c' \in \psi\}.$$

Call algorithm A on the two pairs of forms

$$\text{(A.1)}\ (\varphi_1,\, \psi_0 \wedge \psi_1) \quad \text{and} \quad \text{(A.2)}\ (\psi_1,\, \varphi_0 \wedge \varphi_1)$$

If both calls return `yes`, then return `yes` (as $f = g^d$), otherwise we obtain w such that $f(w) \neq g^d(w)$ in polynomial time (cf. [37]).

Table 1: The first of the two algorithms by Fredman and Khachiyan for deciding duality

The quasi-polynomial time results of Fredman and Khachiyan show that DUAL is most likely not co-NP-complete, which partially answers the question for the complexity status of DUAL. Another important aspect is that, in complexity terms, the class of problems solvable in quasi-polynomial time is closed under quasi-polynomial time transformations; i.e., any problem which is transformable in time $O(n^{\text{polylog}(n)})$ to some problem in this class is solvable in time $O(n^{\text{polylog}(n)})$ as well (such computations compose). Thus in particular, by means of the Fredman-Khachiyan algorithms, dualization serves as a host for proving that problems can be solved in quasi-polynomial time. This has been exploited, e.g., in [11, 9, 15, 11, 16, 17, 18, 19, 32].

5 Follow-Up Work

The seminal paper by Fredman and Khachiyan turned out to be a fruitful basis for analyzing other aspects of the monotone dualization problem, and for considering possible generalizations.

5.1 Computational aspects

The following results have been obtained on different computational aspects of monotone dualization.

Other forms of representation. Gurvich and Khachiyan [46] considered the generation of a prime CNF φ and/or a prime DNF ψ of a monotone function f for various representations of f. They and independently Bioch and Ibaraki [7] showed that generating both φ and ψ simultaneously is feasible in incremental quasi-polynomial time, provided that $f(x)$ can be evaluated in polynomial time. They also showed that this is still feasible in quasi-polynomial total time if $f(x)$ can be evaluated in quasi-polynomial time. On the other hand, generating only φ or ψ is not feasible in polynomial total time (unless $\mathrm{P} = \mathrm{NP}$) for a variety of representations (cf. also [29, 27]).

In further work, Boros et al. extended the quasi-polynomial time results of [37, 46] to generating incrementally the minimal true points of a monotone function f, i.e., minimal vectors x such that $f(x) = 1$, where f is given by a polynomial-time oracle, provided that the set M_f of all minimal true points satisfies a uniform dual-boundedness property [9, 17]. This property requests that the hypergraph $\mathcal{M}_f = (V, \mathcal{E}_f)$ with vertices $V = \{x_1, \ldots, x_n\}$ and an edge $\{x_i \mid v_i = 1\}$ for each $v \in M_f$, is such that for each sub-hypergraph $\mathcal{H} = (V, \mathcal{E})$, where $\mathcal{E} \subseteq \mathcal{E}_f$ is nonempty, the number of joint maximal independent sets of $\mathcal{H}$ and $\mathcal{M}_f$ is quasi-polynomially bounded in the size of $\mathcal{H}$, $|V|$, and the size of the oracle representing f; monotone dualization is covered as a special case, as well as polymatroid functions [11], 2-monotonic functions [15], weighted transversals [17, 19], and others [18]; For more details, we refer to [9].

Limited nondeterminism. Eiter, Gottlob, and Makino [31] and independently Kavvadias and Stavropoulos [52] proved that monotone dualization can be solved with *limited nondeterminism*, i.e., by a polynomial-time algorithm which makes at most poly-logarithmically many nondeterministic steps in the computation (for a survey on limited nondeterminism, see [41]). More precisely, let $g(n)$-P denote the class of problems solvable by a Turing machine in polynomial time with at most $g(n)$ nondeterministic steps (i.e., bit guesses) and $\beta_k\mathrm{P} = \bigcup_c (c \log^k n)$-P for integer $k \geq 1$. As shown in [31], monotone duality is in co-β_2P (and in fact decidable with $O(\chi(n) \cdot \log n)$ many nondeterministic steps). The key to this result is a careful analysis of the information which is needed to describe the way from the root of the recursive computation tree to a node which disproves duality of the input pair (f, g). The crucial observation is here that "left" and "right" children do not occur with the same frequency, and that one may have an exponential gain by counting *how many times in a row* one takes the left child (resp., the right child) rather than using one bit for each step.

While [31] used the original algorithms of Fredman and Khachiyan, Kavvadias and Stavropoulos present in [52] a non-deterministic algorithm which, based on the methods of [37] proceeds in alternating phases of deterministic and non-deterministic computation until the original problem is reduced to constant size. In each deterministic step, $O(n \log n)$ subproblems are generated, of which one is non-deterministically selected for further checking.

Probabilistic aspects. Several papers have considered probabilistic aspects of monotone dualization. Shmulevich et al. [75] showed that almost all monotone Boolean functions are polynomially learnable with membership queries only, if the number of variables n goes to infinity; this result implies that, in probabilistic terms, monotone dualization can be solved by an algorithm with expected polynomial-total running time if

n goes to infinity, if all monotone Boolean functions are equiprobable (and only in few cases, the running time is super-polynomial).

Recently, Gaur and Krishnamurti [40] showed that problem SELF-DUALITY for monotone CNFs, formulated as a special satisfiability problem, is solvable by an algorithm in polynomial time on average over random instances, which uses case decomposition ideas similar to those in the Fredman-Khachiyan algorithms. Furthermore, they showed that a simple basic version of their algorithm decides SELF-DUALITY in time $O(n^{2\log n+2})$.

Work space requirements. The original algorithms by Fredman and Khachiyan in [37] require workspace super-polynomial in the input size, and are thus infeasible from a memory point of view, since the workspace available in practice may be exceeded. Tamaki [79] has improved the enumeration of the dual clauses, based on decomposition methods used in Algorithm B, such that it runs in quasi-polynomial total time (more precisely, in time $(n+m)^{O(\log n)}$, where $n = \|\varphi\|$ and $m = |\psi|$) and in $O(n\log^2 n)$ space. His algorithm uses lexicographic ordering of clauses in order to prevent multiple output of the same clause. By this together with the result in [7], we have an algorithm with incremental quasi-polynomial delay and (input) polynomial space. The randomized improvement of Algorithm A in [10] has an expected total memory requirement of all nodes on a path which is linear in the length $m = \|\varphi\| + \|\psi\|$ of φ and ψ.

5.2 Generalizations to Posets

Fredman and Khachiyan's algorithms have been generalized from the Boolean case to certain classes of partially ordered sets (posets). Here, the problem considered is to recognize respectively to generate the dual of a give set of elements $\mathcal{A}$ in the poset $(P, \preceq)$, which consists of the maximal independent elements of $\mathcal{A}$, i.e., the largest elements in $P \setminus \{b \in P \mid a \preceq b \text{ for some } a \in \mathcal{A}\}$ with respect to $\preceq$. Integer boxes [15], products of forests [36], and products of lattices [35] fall into such generalizations. Together with a dual-boundedness property [9], they imply incremental quasi-polynomial solvability of many generation problems such as generating maximal feasible points of monotone linear systems, inefficient points of probability distributions, and maximal boxes (e.g., [15, 16]).

6 Other Algorithms

Apart from the algorithms discussed in the previous sections, a number of further algorithms for dualization have been proposed. In this section, we consider algorithms which are based on methods in learning theory, on advanced partitioning, and on parallel computation, respectively.

Learning-based algorithms. Some important task in concept learning is to compute the boundary, given by the set $\min(T)$ of the minimal true vectors and the set $\max(F)$ of maximal false vectors of a monotone Boolean function f with membership queries, i.e., an oracle for deciding whether $v \in T$ holds for a given vector v . It has been shown that this problem is polynomial-time equivalent to monotone dualization [7]. This result can be exploited to derive polynomial-time solvability of monotone dualization, thanks to results about learning, for several classes of functions (e.g., [20, 24, 26, 64]).

We note that computing only minimal true vectors (resp., maximal false vectors) is hard and requires exponential time by information-theoretic bounds. It is known [1] that monotone DNFs (or CNFs) are not exact learnable with membership oracles alone in time polynomial in the size of the target DNF (or CNF) formula, since information theoretic barriers entail a lower bound of $|\mathrm{CNF}(f)| + |\mathrm{DNF}(f)|$ for the number of queries which are needed to learn a monotone function f.

Algorithms for computing the boundary based on chain decomposition of the search space, have been proposed, following different query strategies, by Hansel [47] and by Sokolov [77]; Gainanov [38] proposed an algorithm that constructs border vectors one by one, by making at most $n + 1$ queries for each vector starting from a yet unclassified vector, where n is the number of variables. The same method was applied in [20, 63, 84]. Finding an unknown vector for a function is not easy, and is in fact equivalent to monotone dualization. One may solve this problem by a neighborhood search algorithm in an already known border. While this method is exponential in general, it works well in several cases [63]. Such an algorithm is also used to derive the result that almost all monotone Boolean functions are learnable using membership queries in polynomial time [75], which implies that almost all monotone Boolean functions are dualizable in polynomial time.

Ordered variable-decomposition algorithms. In this subsection, we regard DUALIZATION as the problem of computing $\mathrm{DNF}(f)$ from $\mathrm{CNF}(f)$, where f is monotone (recall that a prime CNF of the dual function f^d is easily obtained from $\mathrm{DNF}(f)$). Let

$$\varphi \;=\; c_1 \wedge \cdots \wedge c_m \tag{8}$$

be the prime CNF of f, where we assume without loss of generality that all variables x_j ($j = 1, 2, \ldots n$) appear in φ. Let φ_i ($i = 0, 1, \ldots, n$) be the CNF obtained from φ by fixing variables $x_j = 1$ for all j with $j \geq i + 1$. By definition, we have $\varphi_0 = \top$ (truth) and $\varphi_n = \varphi$. For example, consider $\varphi = (x_1 \vee x_2)(x_1 \vee x_3)(x_2 \vee x_3 \vee x_4)(x_1 \vee x_4)$. Then we have $\varphi_0 = \varphi_1 = \top$, $\varphi_2 = (x_1 \vee x_2)$, $\varphi_3 = (x_1 \vee x_2)(x_1 \vee x_3)$, and $\varphi_4 = \varphi$. Similarly, for the prime DNF

$$\psi \;=\; t_1 \vee \cdots \vee t_k \tag{9}$$

of f, we denote by ψ_i the DNF obtained from ψ by fixing variables $x_j = 1$ for all j with $j \geq i+1$. Clearly, we have $\varphi_i \equiv \psi_i$, i.e., φ_i and ψ_i represent the same function denoted by f_i.

Denote by Δ^i ($i = 1, 2, \ldots, n$) the CNF consisting of all the clauses in φ_i but not in φ_{i-1}. For the above example, we have $\Delta^1 = 1$, $\Delta^2 = (x_1 \vee x_2)$, $\Delta^3 = (x_1 \vee x_3)$, and $\Delta^4 = (x_2 \vee x_3 \vee x_4)(x_1 \vee x_4)$. Note that $\varphi_i = \varphi_{i-1} \wedge \Delta^i$; hence, for all $i = 1, 2, \ldots, n$ we have

$$\psi_i \equiv \psi_{i-1} \wedge \Delta^i \;\equiv \bigvee_{t \in \mathrm{DNF}(f_{i-1})} (t \wedge \Delta^i). \tag{10}$$

Let $\Delta^i[t]$, for $i = 1, \ldots, n$ denote the CNF consisting of all the clauses c such that c contains no literal in t_{i-1} and $c \vee x_i$ appears in Δ^i. For example, if $t = x_2x_3x_4$ and $\Delta^4 = (x_2 \vee x_3 \vee x_4)(x_1 \vee x_4)$, then $\Delta^4[t] = x_1$. It follows from (10) that for all $i = 1, 2, \ldots, n$

$$\psi_i \equiv \bigvee_{t \in \mathrm{DNF}(f_{i-1})} \Big((t \wedge \Delta^i[t]) \;\vee\; (t \wedge x_i) \Big). \tag{11}$$

It is not difficult to see that $\Delta^i[t] \leq |\mathrm{DNF}(f_{i-1})| \leq |\psi|$ holds for all i and $t \in \mathrm{DNF}(f_{i-1})$. Different from clause-based decomposition in Section 3, the intermediate results are always bounded by the output length and hence this scheme is efficient if one can compute all $\Delta^i[t]$'s efficiently.

This scheme was used earlier in the graph case by Tsukiyama et al. [81], and later for hypergraphs by Lawler et al. [57]. By means of it, several results have been obtained.

- A straight implementation of this scheme yields an algorithm which shows that monotone dualization is solvable in polynomial total time for many well-known classes of functions, including functions that are degenerated, read-bounded, acyclic, or have bounded treewidth (under different notions) [31], where the bound is non-constant. On the other hand, the problem is as hard as in the general even for hypertree-width 2 [30]. An incremental polynomial-time algorithm can be constructed from the scheme, provided that the function is minor-closed [7]. However, while polynomial this algorithm is rather slow.

- Exploiting the scheme in a depth-first manner leads to a dualization algorithm which runs in polynomial space (in the size of the input). However, this algorithm will not be fast, since essentially we trade space for time by it.

- A more sophisticated implementation using priority queues yields a speedup and enables enumeration according to an ordering (which, moreover, is often polynomial delay); it may use exponential work space in the input size, though [31].

Parallel algorithms. As for parallel computation, two types of problems have been considered around the recognition problem DUAL and the generation problem DUALIZATION, respectively. The first type is generating a counterexample to duality based on a dual subclauses (subtransversals), i.e., sets of variables included in some dual clause [8, 14]. In general, deciding whether a clause is a dual subclause is NP-complete, but for certain classes of functions it is polynomial (e.g., k-CNF, k-conformal functions). This is then exploited for parallelization.

The second type is generating some resp. all dual clauses. While a single clause in the dual form of a given CNF is efficiently computable in parallel for certain classes of CNFs, it is open whether this is possible for all CNFs, cf. [50]. The papers [13, 55] considers parallel generation of a given number k of clauses in the dual form of a CNF φ. As shown there, this problem can be solved in time $O(\delta \log(1+k)\mathrm{polylog}(n\delta))$ using $n^{O(\log k)}k^{O(\delta)}$ many processors if φ is *uniformly δ-sparse*, i.e., for every set of variables $S \subseteq V$, the average variable degree in the CNF $\varphi_S = \{c \in \varphi \mid c \subseteq S\}$ is bounded by δ. This implies that the problem is in NC whenever δ is bounded by a constant (e.g., if φ is read-bounded). Exploiting this result, also new results on the complexity of dualization for new classes of CNFs corresponding to hypergraphs of bounded dual-conformality, and hypergraphs in which every edge intersects every minimal transversal in a bounded number of vertices, were derived.

Several other algorithms exist which have been proposed in different domains (cf. [23, 27, 66, 44]); in particular, in the area of diagnosis, several algorithms have been considered, cf. [71, 42, 85, 59].

7 Implementations and Experiments

Many algorithms for solving the dualization problem or equivalent problems in applications have been implemented. Often these algorithms are (slight) variants of the algorithms described in the literature.

Boros et al. [10] report about an implementation of a randomized variant of Algorithm A in [37] (in fact, their algorithm solves the more general problem of generating all maximal independent elements of a set of vectors from an integral box). The algorithm randomly guesses an additional dual clause (making up to a fixed number of tries), and delays the actual computation of recursive inputs. The expected number of recursive calls is shown to be $nm^{O(\log^2 m)}$, where m is $|\varphi|$ plus the number of clauses produced so far. The experiments reported on randomly generated and designed inputs indicate that using randomization offers substantial improvements, and that for random inputs the average CPU time per transversal does not increase more than linearly with increasing the number of variables or clauses.

Bailey et al. [3] present a hybrid of an vertex-based and an edge-based algorithm for dualization. Informally, the variables are ordered by increasing frequency $x_{i_1}, x_{i_2}, x_{i_3} \ldots$, and then the dual clauses are generated by sprouting as in Section 3. This scheme is recursively applied unless the volume of an instance is below a certain threshold; in this case, an optimized version of the Berge algorithm is employed. Experimental results on instances which are randomly generated from machine learning data sets show that this algorithm is faster than others, including Algorithm B in [37] (which is significantly slower), the Kavvadias-Stavropoulos algorithm [51], and algorithms proposed in the data mining domain.

Kavvadias and Stavropoulos [51, 53] present a variant of the Berge algorithm which makes two main improvements. Firstly, variables which occur in exactly the same clauses are recursively factored out, by replacing them with a new variable. In the obtained dual clauses, the latter is substituted back with any of these variables. Secondly, the generation of clauses is performed depth-first rather than breadth-first; in this way, the time until the first output can be polynomially bounded, while under usual breadth-first processing it is exponential in general. A further improvement assures that no regeneration of minimal transversals occurs at any intermediate level (this aim is similar as in [82]). Like Uno's algorithm [82], it uses only space polynomial in the input size. The experimental results reported in [51] for randomly generated inputs show that the algorithm behaves very well compared with a naive dualization algorithm, and that, interestingly, the time between subsequent outputs is quite uniform. Another finding was that an efficient implementation of sets (as in the work reported, by bitmaps) is an important factor. In [53], Kavvadias and Stavropoulos compare their algorithm (with some advances in implementation) against the algorithms of Boros et al. (BEGK) [10] and Bailey et al. (BMR) [3], on the test sets from [10, 3] and on random instances. On the data sets from [10], the Kavvadias-Stavropoulos (KS) algorithm outperforms the other two with respect to runtime, with the BMR algorithm being the second fastest in most cases. However, in several test cases, some other algorithm uses less memory. In particular, it seems that the KS algorithm behaves better if the dual CNF has more clauses than the input CNF. On the data sets from [3], the BEGK and BMR algorithms [10, 3] are faster, and the BMR algorithm uses the least memory. On the random instances considered, the runtimes of the algorithms greatly vary, but the KS algorithm uses far less memory than the other ones.

Uno [82] presents another variant of the Berge algorithm, in which the fact is exploited that every variable x in every dual clause c of φ must be critical, i.e., there must be some clause $c' \in \varphi$ such that $c \cap c' = \{x\}$. In this way, the dual clauses can be uniquely generated in a computation tree. As reported in [83], this tree was in the experiments linear in the size of φ^d, even though it still may have exponential intermediate

size. Further improvements reduce the effort for criticality checking and the number of iterations, by keeping a set of uncovered clauses. It is reported in [83] that in experiments (comprising data from frequent-set problems in knowledge discovery) the memory requirement was linear in the input size, compared with a $O|n{\cdot}\varphi|)$ worst case bound (which can be improved upon) where n is the number of variables.

Lin and Jiang [59] report experimental results for the HS-tree algorithm from [71], and for different implementations of (slight variants of) the vertex-based decomposition algorithm outlined in Section 3, named BHS-tree algorithm and Boolean algorithm. According to their results, the management of trees data structures as in HS-tree and BHS-tree has significant overhead compared with the Boolean algorithm, which was implemented using lists.

Interesting results for dualization algorithms in the learning scheme, where the number of membership queries is most significant, have been obtained by Torvik and Triantaphyllou [80], who studied the minimum number of queries which are needed by an optimal algorithm on average, and compared the algorithms by Hansel [47], Sokolov [77], Gainanov [38] (known as FINDBORDER), and a novel algorithm, which is artificially designed and less intuitive, against it. Using an unbiased sampling framework, they found that the popular FINDBORDER algorithm makes almost twice as many queries compared with the minimum, and that, interestingly, the earliest algorithm [47] is best among those in the previous literature. For the average number of queries to identify each border vector of f, i.e., minimal (resp. maximal) vector v such that $f(v) = 1$ (resp., $f(v) = 0$), we have a similar picture. FINDBORDER needs about two queries (for an optimum of one), while Hansel's algorithm performed best among those from the previous literature; the novel algorithm in [80] outperforms all previous known algorithms.

8 Discussion and Conclusion

We have briefly surveyed work that deals with computational aspects of monotone dualization, focusing on the important Fredman-Khachiyan paper [37] and follow-up papers. While significant progress has been made on the monotone dualization problem over the last decade, there are still many issues open. The most important issue concerning the computational complexity is, of course, whether the problem (in its decisional variant DUAL) is solvable in polynomial time or not. In line with resolving this question, pushing the tractability frontier for the problem by identifying new interesting polynomial classes of dualization remains an important issue. One example of such an interesting class for which this is open is $\log$-CNFs (i.e., CNFs with clauses of logarithmic size).

Other complexity issues are solvability of DUAL in polylog space, and the issue of lower bounds for the problem. It is, for instance, yet unknown whether monotone dualization is P-hard or hard for nondeterministic logspace (under logspace reductions). As for the parallel complexity of the problem, it is open whether the generation problem DUALIZATION can be solved in polylog-time (in the input length) with quasi-polynomial many processors (in the combined size of the input and output).

On the algorithms side, while there are studies of the behavior of some algorithms for monotone dualization (e.g., [51, 83, 10, 3, 59, 80]), a comprehensive, systematic experimental evaluation of the many algorithms for this problem is still missing to date. However, it seems that the more refined algorithms do no show a blatantly exploding behavior, as typical with intractable problems, on instances occurring in practice. Along with the experimental evaluation, an interesting task is to compile a collection of "hard" instances of dualization for different (types of) algorithms.

An interesting issue are non-monotone dualization problems which are polynomial-time equivalent to monotone dualization. Here, several interesting results have been obtained. Khardon [56] showed that generating the set of all prime implicants of a given Horn CNF φ is polynomial-time equivalent to monotone dualization. While this set, viewed as a DNF ψ, is equivalent to φ, it is redundant in general and a sub-DNF $\psi' \subset \psi$ may equivalently represent ψ; so, one is interested in a non-redundant such ψ' for representing φ. While the problem of generating any nonredundant ψ' is at least as hard as monotone dualization, it is open whether this problem is polynomial-time equivalent to it or harder [56]. For the class of bidual Horn functions (where φ^d represents also a Horn function), this problem is equivalent to monotone dualization [34]; interestingly, bidual Horn functions have, like monotone functions, a unique non-redundant prime DNF.

Another interesting issue is the Horn transformation problem between Horn CNFs and characteristic sets, which are unique representations of Horn functions. It is known that the transformation problem between Horn CNFs consisting of all prime implicates and characteristic sets is polynomial-time equivalent to monotone dualization, while it is open for nonredundant Horn CNFs [56].

Further issues are generating the dual form under certain restrictions, or producing only a part of the dual form efficiently. Instances of the first kind are enumerating the clauses of the dual function with bounded delay, or under work space constraints. For example, for quadratic CNFs the dual clauses can be enumerated with polynomial delay and in (input) polynomial space [81, 65], and with polynomial delay if in addition lexicographic ordering is required [49]. It is open whether the polynomial delay and the polynomial space result extends to k-CNFs, for constant $k \geq 3$. In the general case, it would be interesting to know whether the dual clauses can be enumerated with quasi-polynomial time delay in the input size or, a bit weaker, with amortized such delay (i.e., delay $mn^{O(\log n)}$ where $m = |\psi|$ and $n = \|\varphi\|$) [79].

Finally, producing only a part of the dual form can be viewed as a generalization of the problem; an instance of this is enumerating all dual clauses of bounded size, cf. [22]. Note that generating a given number k of clauses in the dual form is (under a suitable notion) polynomial-time equivalent to DUAL. The problem is in RNC (more precisely, solvable in time $\mathrm{polylog}(n, |\varphi|, k)$ on a number of processors polynomial in n and $|\varphi|$) if φ has bounded clause size [12] as well as for other classes of formulas [55].

Acknowledgments

We are grateful to Hisao Tamaki for sending us an extended version of [79].

References

[1] D. Angluin. Queries and concept learning. *Machine Learning*, 2:319–342, 1996.

[2] D. Angluin, M. Frazier, and L. Pitt. Learning conjunctions of Horn clauses. *Machine Learning*, 9:147–164, 1992.

[3] 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), 19-22 December 2003, Melbourne, Florida, USA*, pages 485–488. IEEE Computer Society, 2003.

[4] C. Benzaken. Algorithme de dualisation d'une fonction booléenne. *Revue Francaise de Traitment de l'Information – Chiffres*, 9(2):119–128, 1966.

[5] C. Berge. *Hypergraphs*, volume 45 of *North Holland Mathematical Library*. Elsevier Science Publishers B.V. (North-Holland), 1989.

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

[7] C. Bioch and T. Ibaraki. Complexity of identification and dualization of positive Boolean functions. *Information and Computation*, 123:50–63, 1995.

[8] E. Boros, K. M. 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(4):253–266, 2000.

[9] E. Boros, K. M. Elbassioni, V. Gurvich, and L. Khachiyan. Generating dual-bounded hypergraphs. *Optimization Methods and Software*, 17(5):749 – 781, 2002.

[10] E. Boros, K. M. Elbassioni, V. Gurvich, and L. Khachiyan. An efficient implementation of a quasi-polynomial algorithm for generating hypergraph transversals. In G. D. Battista and U. Zwick, editors, *Proceedings 11th Annual European Symposium on Algorithms (ESA 2003), Budapest, Hungary, September 16-19, 2003*, volume 2832 of *Lecture Notes in Computer Science*, pages 556–567. Springer, 2003.

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

[12] E. Boros, K. M. Elbassioni, V. Gurvich, and L. Khachiyan. A global algorithm for finding all minimal transversals of hypergraphs of bounded edge-size. Technical Report DIMACS 2004-31, Rutgers University, 2004.

[13] E. Boros, K. M. Elbassioni, V. Gurvich, and L. Khachiyan. Computing many maximal independent sets for sparse hypergraphs in parallel. Technical Report RUTCOR RRR 33-2004, Rutgers University, Oct. 2004.

[14] E. Boros, K. M. Elbassioni, V. Gurvich, and L. Khachiyan. Generating maximal independent sets for hypergraphs with bounded edge-intersections. In M. Farach-Colton, editor, *Proceedings of the 6th Latin American Symposium on Theoretical Informatics (LATIN 2004), Buenos Aires, Argentina, April 5-8, 2004*, volume 2976 of *Lecture Notes in Computer Science*, pages 488–498. Springer, 2004.

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

[16] E. Boros, K. M. Elbassioni, V. Gurvich, L. Khachiyan, and K. Makino. An intersection inequality for discrete distributions and related generation problem. In *Proceedings of the 30th International Colloquium on Automata, Languages and Programming (ICALP 2003), Eindhoven, The Netherlands, June 30 - July 4, 2003*, volume 2719 of *Lecture Notes in Computer Science*. Springer, 2003.

[17] E. Boros, V. Gurvich, L. Khachiyan, and K. Makino. Dual-bounded generating problems: Partial and multiple transversals of a hypergraph. *SIAM Journal on Computing*, 30(6):2036–2050, 2000.

[18] E. Boros, V. Gurvich, L. Khachiyan, and K. Makino. On maximal frequent and minimal infrequent sets in binary matrices. *Annals of Mathematics and Artificial Intelligence*, 39(3):211–221, 2003.

[19] E. Boros, V. Gurvich, L. Khachiyan, and K. Makino. Dual-bounded generating problems: weighted transversals of a hypergraph. *Discrete Applied Mathematics*, 142(1-3):1–15, 2004.

[20] E. Boros, P. Hammer, T. Ibaraki, and K. Kawakami. Polynomial time recognition of 2-monotonic positive Boolean functions given by an oracle. *SIAM J. Comput.*, 26:93–109, 1997.

[21] Y. Crama. Dualization of regular boolean functions. *Discrete Applied Mathematics*, 16:79–85, 1987.

[22] P. Damaschke. Parameterized enumeration, transversals, and imperfect phylogeny reconstruction. In R. G. Downey, M. R. Fellows, and F. K. H. A. Dehne, editors, *Proceedings First International Workshop on Parameterized and Exact Computation (IWPEC 2004), Bergen, Norway, September 14-17, 2004*, volume 3162 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2004.

[23] J. Demetrovics and V. D. Thi. Keys, antikeys and prime attributes. *Annales Univ. Sci. Budapest, Sect. Comp.*, 8:35–52, 1987.

[24] C. Domingo. Exact learning of subclasses of cdnf formulas with membership queries. In J. yi Cai and C. K. Wong, editors, *Proceedings 2nd Annual International Conference on Computing and Combinatorics (COCOON '96), Hong Kong, June 17-19, 1996*, volume 1090 of *Lecture Notes in Computer Science*, pages 179–188. Springer, 1996.

[25] C. Domingo. Polynominal time algorithms for some self-duality problems. In G. C. Bongiovanni, D. P. Bovet, and G. D. Battista, editors, *Proceedings Third Italian Conference on Algorithms and Complexity (CIAC '97), Rome, Italy, March 12-14, 1997*, volume 1203 of *Lecture Notes in Computer Science*, pages 171–180. Springer, 1997.

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

[27] T. Eiter. *On Transversal Hypergraph Computation and Deciding Hypergraph Saturation*. PhD thesis, Institut für Informationssysteme, TU Wien, Austria, October 1991.

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

[29] T. Eiter and G. Gottlob. Identifying the minimal transversals of a hypergraph and related problems. *SIAM Journal on Computing*, 24(6):1278–1304, December 1995. Extended paper Technical Report CD-TR 91/16, Christian Doppler Laboratory for Expert Systems, TU Vienna, Austria, January 1991.

[30] T. Eiter and G. Gottlob. Hypergraph transversal computation and related problems in logic and ai. In S. Flesca et al., editor, *Proceedings 8th European Conference on Logics in Artificial Intelligence – Journées Européennes sur la Logique en Intelligence Artificielle (JELIA 2002), Cosenza, Italy, 2002*, number 2424 in LNCS, pages 549–564. Springer, 2002.

[31] T. Eiter, G. Gottlob, and K. Makino. New Results on monotone dualization and generating hypergraph transversals. *SIAM Journal on Computing*, 32(2):514–537, January-February 2003. Preliminary paper in Proc. ACM STOC 2002.

[32] T. Eiter and K. Makino. Abduction and the dualization problem. In G. Grieser, Y. Tanaka, and A. Yamamoto, editors, *Proceedings 6th International Conference on Discovery Science (DS 2003), Sapporo, Japan, October 17-19, 2003*, volume 2843 of *LNCS/LNAI*, pages 1–20. Springer, 2003.

[33] T. Eiter, T. Ibaraki, and K. Makino. Double Horn functions. *Information and Computation*, 144:155–190, August 1998.

[34] T. Eiter, T. Ibaraki, and K. Makino. Bidual Horn functions and extensions. *Discrete Applied Mathematics*, 96-97:55–88, 1999. Selected for the special volume *Discrete Applied Mathematics, Editors' Choice, Edition 1999*.

[35] K. M. Elbassioni. An algorithm for dualization in products of lattices and its applications. In R. H. Möhring and R. Raman, editors, *Proceedings 10th Annual European Symposium on Algorithms (ESA 2002), Rome, Italy, September 17-21, 2002*, volume 2461 of *Lecture Notes in Computer Science*, pages 424–435. Springer, 2002.

[36] K. M. Elbassioni. On dualization in products of forests. In H. Alt and A. Ferreira, editors, *Proceedings 19th Annual Symposium on Theoretical Aspects of Computer Science (STACS 2002), Antibes - Juan les Pins, France, March 14-16, 2002*, volume 2285 of *Lecture Notes in Computer Science*, pages 142–153. Springer, 2002.

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

[38] D. N. Gainanov. On one criterion of the optimality of an algorithm for evaluating monotonic Boolean functions. *U.S.S.R. Computational Mathematics and Mathematical Physics*, 24:176–184, 1984.

[39] D. R. Gaur and R. Krishnamurti. Self-duality of bounded monotone Boolean functions and related problems. In H. Arimura, S. Jain, and A. Sharma, editors, *Proceedings 11th International Conference on Algorithmic Learning Theory (ALT 2000), Sydney, Australia, December 11-13, 2000*, volume 1968 of *Lecture Notes in Computer Science*, pages 209–223. Springer, 2000.

[40] D. R. Gaur and R. Krishnamurti. Average case self-duality of monotone Boolean functions. In A. Y. Tawfik and S. D. Goodwin, editors, , *Proceedings 17th Conference of the Canadian Society for Computational Studies of Intelligence on Advances in Artificial Intelligence, (Canadian AI 2004), London, Ontario, Canada, May 17-19, 2004*, volume 3060 of *Lecture Notes in Computer Science*, pages 322–338. Springer, 2004.

[41] J. Goldsmith, M. Levy, and M. Mundhenk. Limited nondeterminism. *SIGACT News*, 27(2):20–29, June 1996.

[42] R. Greiner, B. A. Smith, and R. W. Wilkerson. A correction to the algorithm in Reiter's theory of diagnosis. *Artificial Intelligence*, 41:79–88, 1990.

[43] A. Grossi. Algorithme á seperation de variables pour la dualisation d'une fonctione booléenne. *Revue Francaise D'Automatique, Informatique et Recherche*, B-1:41–55, 1974.

[44] D. Gunopulos, R. Khardon, H. Mannila, and H. Toivonen. Data mining, hypergraph transversals, and machine learning. In *Proceedings of the 16th ACM SIGACT SIGMOD-SIGART Symposium on Principles of Database Systems (PODS-96)*, pages 209–216, 1993.

[45] V. Gurvich and L. Khachiyan. On the frequency of the most frequently occurring variable in dual monotone dnfs. *Discrete Applied Mathematics*, 169(1-3):245–248, 1997.

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

[47] G. Hansel. Sur le nombre des fonctions booléennses monotones de n variables. In *Comptes Rendus des Séances de l'Académie des Sciences*, volume 248, pages 3522–3523. Paris, Gauthier-Villars, 1959.

[48] D. S. Johnson. Open and closed problems in **NP**-completeness. Lecture given at the International School of Mathematics "G. Stampacchia": Summer School "**NP**-Completeness: The First 20 Years", Erice (Sicily), Italy, 20 - 27 June 1991.

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

[50] R. M. Karp and A. Wigderson. A fast parallel algorithm for the maximal independent set problem. *Journal of the ACM*, 32(4):762–773, 1985.

[51] D. J. Kavvadias and E. C. Stavropoulos. Evaluation of an algorithm for the transversal hypergraph problem. In J. S. Vitter and C. D. Zaroliagis, editors, *Proceedings 3rd International Workshop on Algorithm Engineering (WAE '99), London, UK, July 19-21, 1999*, volume 1668 of *Lecture Notes in Computer Science*, pages 72–84. Springer, 1999.

[52] D. J. Kavvadias and E. C. Stavropoulos. Monotone Boolean dualization is in co-NP[$\log^2$n]. *Information Processing Letters*, 85:1–6, 2003.

[53] D. J. Kavvadias and E. C. Stavropoulos. An efficient algorithm for the transversal hypergraph generation. *Journal of Graph Algorithms and Applications*, 9:239–264, 2005.

[54] A. Kean and G. Tsiknis. An incremental method for generating prime implicants. *Journal of Symbolic Computation*, 9:185–206, 1990.

[55] L. Khachiyan, E. Boros, K. M. Elbassioni, and V. Gurvich. A new algorithm for the hypergraph transversal problem. In L. Wang, editor, *Proceedings 11th Annual International Conference on Computing and Combinatorics (COCOON 2005), Kunming, China, August 16-29, 2005*, volume 3595 of *Lecture Notes in Computer Science*, pages 767–776. Springer, 2005.

[56] R. Khardon. Translating between Horn representations and their characteristic models. *Journal of Artificial Intelligence Research*, 3:349–372, 1995.

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

[58] E. L. Lawler. Covering problems: Duality relations and a new method of solution. *SIAM J. Appl. Math.*, 14(5):1115–1132, 1966.

[59] L. Lin and Y. Jiang. The computation of hitting sets: Review and new algorithms. *Information Processing Letters*, 86(4):177–184, 2003.

[60] L. Lovász. Coverings and colorings of hypergraphs. In *Proceedings of the 4th Conference on Combinatorics, Graph Theory, and Computing*, pages 3–12, 1973.

[61] K. Maghout. Sur la détermination des nombres de stabilité et du nombre chromatique d'un graphe. In *Comptes Rendus des Séances de l'Académie des Sciences*, volume 248, pages 3522–3523. Paris, Gauthier-Villars, 1959.

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

[63] K. Makino and T. Ibaraki. The maximum latency and identification of positive Boolean functions. *SIAM J. Comput.*, 26:1363–1383, 1997.

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

[65] K. Makino and T. Uno. New algorithms for enumerating all maximal cliques. In T. Hagerup and J. Katajainen, editors, *Proceedings 9th Scandinavian Workshop on Algorithm Theory on Algorithm Theory (SWAT 2004), Humlebaek, Denmark, July 8-10, 2004*, volume 3111 of *Lecture Notes in Computer Science*, pages 260–272. Springer, 2004.

[66] H. Mannila and K.-J. Räihä. Algorithms for inferring functional dependencies from relations. *Data and Knowledge Engineering*, 12(1):83–99, 1994.

[67] N. Mishra and L. Pitt. Generating all maximal independent sets of bounded-degree hypergraphs. In *Proceedings enth Annual Conference on Computational Learning Theory (COLT)*, pages 211–217, 1997.

[68] C. Papadimitriou. NP-completeness: A retrospective. In *Proceedings 24th International Colloquium on Automata, Languages, and Programming (ICALP '97)*, LNCS 1256, pages 2–6, 1997.

[69] U. Peled and B. Simeone. Polynomial-time algorithms for regular set-covering and threshold Synthesis. *Discrete Applied Mathematics*, 12:57–69, 1985.

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

[71] R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32:57–95, 1987.

[72] B. Roy. *Algébre moderne et Thèorie des graphes*, volume II. Dunod, Paris, 1970.

[73] R. Rymon. An SE-tree-based prime implicant generation algorithm. *Annals of Mathematics and Artificial Intelligence*, 11(1-4):351–366, 1994.

[74] P. D. Seymour. On the two-coloring of hypergraphs. *Quarterly Journal of Mathematics Oxford*, 25:303–312, 1974.

[75] I. Shmulevich, A. D. Korshunov, and J. Astola. Almost all monotone Boolean functions are polynomially learnable using membership queries. *Information Processing Letters*, 79(5):211–213, 2001.

[76] J. Slagle, C. Chang, and R. Lee. A new algorithm for generating prime implicants. *IEEE Transactions on Computers*, C-19(4):304–310, 1970.

[77] N. A. Sokolov. On the optimal evaluation of monotonic Boolean functions. *U.S.S.R. Computational Mathematics and Mathematical Physics*, 22:207–220, 1982.

[78] K. Takata. On the sequential method for listing minimal hitting sets. In *Proceedings Workshop on Discrete Mathematics and Data Mining, 2nd SIAM International Conference on Data Mining, April 11-13, Arlington, Virginia, USA*, 2002.

[79] H. Tamaki. Space-efficient enumeration of minimal transversals of a hypergraph. In *Proceedings 75th SIGAL Conference of the Information Processing Society of Japan*, pages 29–36. IPSJ, 2000. Extended paper available from the author.

[80] V. I. Torvik and E. Triantaphyllou. Minimizing the average query complexity of learning monotone Boolean functions. *INFORMS Journal on Computing*, 14(2):144–174, 2002.

[81] S. Tsukiyama, M. Ide, H. Ariyoshi, and I. Shirakawa. A new algorithm for generating all maximal independent sets. *SIAM Journal on Computing*, 6:505–517, 1977.

[82] T. Uno. A practical fast algorithm for enumerating minimal set coverings. In *Proceedings 83rd SIGAL Conference of the Information Processing Society of Japan, Toyko, March 15, 2002*, pages 9–16. IPSJ, 2002. In Japanese.

[83] T. Uno and K. Satoh. Detailed description of an algorithm for enumeration of maximal frequent sets with irredundant dualization. In B. Goethals and M. J. Zaki, editors, *Proceedings Workshop on Frequent Itemset Mining Implementations (FIMI '03) at ICDM 2003, 19 December 2003, Melbourne, Florida, USA*, volume 90 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2003.

[84] L. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, 1984.

[85] F. Wotawa. A variant of Reiter's hitting-set algorithm. *Information Processing Letters*, 79(1):45–51, 2001.