アルゴリズム 88－6
（2003．1．20）

単調双対化問題とハイパーグラフ横断列挙問題に対する新しい結果について[1]

Thomas Eiter[†], Georg Gottlob[‡], 牧野和久[§]

[†]Institut für Informationssysteme, Technische Universität Wien, Favoritenstraße 9-11, A-1040 Vienna, Austria.
Email: eiter@kr.tuwien.ac.at

[‡]Institut für Informationssysteme, Technische Universität Wien, Favoritenstraße 9-11, A-1040 Vienna, Austria.
Email: gottlob@dbai.tuwien.ac.at

[§]560-8531 大阪府豊中市待兼山町 1-3 大阪大学大学院基礎工学研究科システム科学分野
Email: makino@sys.es.osaka-u.ac.jp

あらまし 我々は単調 CNF を双対化する（すなわち，ハイパーグラフのすべての極小横断を求める）問題を考察する．この問題に関連する決定問題は，NP 完全性に関する有名な未解決問題である．我々は，単調 CNF の重要な様々な部分クラスに対して多項式時間，あるいは，出力多項式時間アルゴリズムを開発する．これらの結果は，単調双対化問題が効率的に解けるクラスを広げ，計算時間等の意味で過去の結果を改善する．さらに，我々は，2 つの単調 CNF の非双対性が制限された非決定性計算で可能であることを示す．より正確には，2 つの単調 CNF の非双対性が $O(\chi(n)\cdot\log n)$ の推定ビットを用いて多項式時間で解けることを示す．ただし，$\chi(n)$ は，$\chi(n)^{\chi(n)}=n$ をみたす数であり，$\chi(n)=o(\log n)$ が成立する．
和文キーワード: 双対化, ハイパーグラフ, 横断計算, 出力多項式アルゴリズム, 組合せ列挙, 木幅, ハイパーグラフの非巡回性, 制限的非決定性

New Results on Monotone Dualization and Generating Hypergraph Transversals

Thomas Eiter[†], Georg Gottlob[‡], and Kazuhisa Makino[§]

[†]Institut für Informationssysteme, Technische Universität Wien, Favoritenstraße 9-11, A-1040 Vienna, Austria.
Email: eiter@kr.tuwien.ac.at

[‡]Institut für Informationssysteme, Technische Universität Wien, Favoritenstraße 9-11, A-1040 Vienna, Austria.
Email: gottlob@dbai.tuwien.ac.at

[§]Division of Systems Science, Graduate School of Engineering Science, Osaka University, Toyonaka, Osaka, 560-8531, Japan. Email: makino@sys.es.osaka-u.ac.jp

Abstract We consider the problem of dualizing a monotone CNF (equivalently, computing all minimal transversals of a hypergraph), whose associated decision problem is a prominent open problem in NP-completeness. We present a number of new polynomial time resp. output-polynomial time results for significant cases, which largely advance the tractability frontier and improve on previous results. Furthermore, we show that duality of two monotone CNFs can be disproved with limited nondeterminism. More precisely, this is feasible in polynomial time with $O(\chi(n)\cdot\log n)$ suitably guessed bits, where $\chi(n)$ is given by $\chi(n)^{\chi(n)}=n$; note that $\chi(n)=o(\log n)$.
英文 key words: Dualization, Hypergraphs, Transversal computation, Output-polynomial algorithms, Combinatorial enumeration, Treewidth, Hypergraph acyclicity, Limited nondeterminism.

[1]Extended abstract of this paper appears in: Proceedings of the 34th ACM Symposium on Theory of Computing (STOC-02), May 19-21, 2002, Montreal, Quebec, Canada.

1 Introduction

Recall that the prime CNF of a monotone Boolean function f is the unique formula $\varphi = \bigwedge_{c \in S} c$ in conjunctive normal form where S is the set of all prime implicates of f, i.e., minimal clauses c which are logical consequences of f. In this paper, we consider the following problem:

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.

Here $f^d = \overline{f}(\overline{x}_1, \ldots, \overline{x}_m)$. It is well known that DUALIZATION is equivalent to the TRANSVERSAL COMPUTATION problem, which requests to compute the set of all minimal transversals (i.e., minimal hitting sets) of a given hypergraph $\mathcal{H}$, in other words, the *transversal hypergraph* $Tr(\mathcal{H})$ of $\mathcal{H}$. Actually, these problems can be viewed as the same problem, if the clauses in a monotone CNF φ are identified with the sets of variables they contain. DUALIZATION is a search problem; the associated decision problem DUAL is to decide whether two given monotone prime CNFs φ and ψ represent a pair (f, g) of dual Boolean functions. Analogously, the decision problem TRANS-HYP associated with TRANSVERSAL COMPUTATION is deciding, given hypergraphs $\mathcal{H}$ and $\mathcal{G}$, whether $\mathcal{G} = Tr(\mathcal{H})$.

DUALIZATION and several problems which are like transversal computation known to be computationally equivalent to problem DUALIZATION (see [14]) are of interest in various areas such as database theory (e.g. [37, 48]), machine learning and data mining (e.g., [5, 6, 11, 21]), game theory (e.g. [25, 41, 42]), artificial intelligence (e.g., [20, 27, 28, 43]), mathematical programming (e.g., [4]), and distributed systems (e.g., [17, 26]) to mention a few.

While the output CNF ψ can be exponential in the size of φ, it is currently not known whether ψ can be computed in *output-polynomial* (or *polynomial total*) *time*, i.e., in time polynomial in the combined size of φ *and* ψ. Any such algorithm for DUALIZATION (or for TRANSVERSAL COMPUTATION) would significantly advance the state of the art of several problems in the above application areas. Similarly, the complexity of DUAL (equivalently, TRANS-HYP) is open since more than 20 years now (cf. [2, 14, 29, 30, 32]).

Note that DUALIZATION is solvable in polynomial total time on a class $\mathcal{C}$ of hypergraphs iff DUAL is in PTIME for all pairs $(\mathcal{H}, \mathcal{G})$, where $\mathcal{H} \in \mathcal{C}$ [2]. DUAL is known to be in co-NP and the best currently known upper time-bound is quasi-polynomial time [16, 18, 46]. Determining the complexities of DUALIZATION and DUAL, and of equivalent problems such as the transversal problems, is a prominent open problem. This is witnessed by the fact that these problems are cited in a rapidly growing body of literature and have been referenced in various survey papers and complexity theory retrospectives, e.g. [29, 33, 39].

Given the importance of monotone dualization and equivalent problems for many application areas, and given the long standing failure to settle the complexity of these problems, emphasis was put on finding tractable cases of DUAL and corresponding polynomial total-time cases of DUALIZATION. In fact, several relevant tractable classes were found by various authors; see e.g. [3, 7, 8, 9, 11, 13, 14, 19, 34, 35, 38, 40] and references therein. Moreover, classes of formulas were identified on which DUALIZATION is not just polynomial total-time, but where the conjuncts of the dual formula can be enumerated with *incremental polynomial delay*, i.e., with delay polynomial in the size of the input plus the size of all conjuncts so far computed, or even with *polynomial delay*, i.e., with delay polynomial in the input size only. On the other hand, there are also results which show that certain well-known algorithms for DUALIZATION are not polynomial-total time. For example, [14, 38] pointed out that a well-known sequential algorithm, in which the clauses c_i of a CNF $\varphi = c_1 \wedge \cdots \wedge c_m$ are processed in order $i = 1, \ldots, m$, is not polynomial-total time in general. Most recently, [45] showed that this holds even if an optimal ordering of the clauses is assumed (i.e., they may be arbitrarily arranged for free).

Main Goal. The main goal of this paper is to present important new polynomial total time cases of DUALIZATION and, correspondingly, PTIME solvable subclasses of DUAL which significantly improve previously considered classes. Towards this aim, we first present a new algorithm DUALIZE and prove its correctness. DUALIZE can be regarded as a generalization of a related algorithm proposed by Johnson, Yannakakis, and Papadimitriou [30]. As other dualization algorithms, DUALIZE reduces the original problem by self-reduction to smaller instances. However, the sub-

division into subproblems proceeds according to a particular order which is induced by an arbitrary fixed ordering of the variables. This, in turn, allows us to derive some bounds on intermediate computation steps which imply that DUALIZE, when applied to a variety of input classes, outputs the conjuncts of ψ with polynomial delay or incremental polynomial delay. In particular, we show positive results for the following input classes:

• **Degenerate CNFs.** We generalize the notion of k-degenerate graphs [49] to hypergraphs and define *k-degenerate monotone CNFs* resp. *hypergraphs*. We prove that for any constant k, DUALIZE works with polynomial delay on k-degenerate CNFs. Moreover, it works in output-polynomial time on $O(\log n)$-degenerate CNFs.

• **Read-k CNFs.** A CNF is *read-k*, if each variable appears at most k times in it. We show that for read-k CNFs, problem DUALIZATION is solvable with polynomial delay, if k is constant, and in total polynomial time, if $k = O(\log(\|\varphi\|))$. Our result for constant k significantly improves upon the previous best known algorithm [11], which has a higher complexity bound, is not polynomial delay, and outputs the clauses of ψ in no specific order. The result for $k = O(\log \|\varphi\|)$ is a non-trivial generalization of the result in [11], which was posed as an open problem [10].

• **Acyclic CNFs.** There are several notions of hypergraph resp. monotone CNF acyclicity [15], where the most general and well-known is α-acyclicity. As shown in [14], DUALIZATION is polynomial total time for β-acyclic CNFs; β-acyclicity is the hereditary version of α-acyclicity and far less general. A similar result for α-acyclic prime CNFs was left open. (For non-prime α-acyclic CNFs, this is trivially as hard as the general case.) In this paper, we give a positive answer and show that for α-acyclic (prime) φ, DUALIZATION is solvable with polynomial delay.

• **Formulas of Bounded Treewidth.** The *treewidth* [44] of a graph expresses its degree of cyclicity. Treewidth is an extremely general notion, and bounded treewidth generalizes almost all other notions of near-acyclicity. Following [12], we define the treewidth of a hypergraph resp. monotone CNF φ as the treewidth of its associated (bipartite) variable-clause incidence graph. We show that DUALIZATION is solvable with polynomial delay (exponential in k) if the treewidth of φ is bounded by a constant k, and in polynomial total time if the treewidth is $O(\log \log \|\varphi\|)$.

• **Recursive Applications of DUALIZE and k-CNFs.** We show that if DUALIZE is applied recursively and the recursion depth is bounded by a constant, then DUALIZATION is solved in polynomial total time. We apply this to provide a simpler proof of the known result [7, 14] that monotone k-CNFs (where each conjunct contains at most k variables) can be dualized in output-polynomial time.

After deriving the above results, we turn our attention to the fundamental computational nature of problems DUAL and TRANS-HYP in terms of complexity theory.

Limited nondeterminism. In a landmark paper, Fredman and Khachiyan [16] proved that problem DUAL can be solved in quasi-polynomial time. More precisely, they first gave an algorithm A solving the problem in $n^{O(\log^2 n)}$ time, and then a more complicated algorithm B whose runtime is bounded by $n^{4\chi(n)+O(1)}$ where $\chi(n)$ is defined by $\chi(n)^{\chi(n)} = n$. As noted in [16], $\chi(n) \sim \log n / \log \log n = o(\log n)$; therefore, duality checking is feasible in $n^{o(\log n)}$ time. This is the best upper bound for problem DUAL so far, and shows that the problem is most likely not NP-complete.

A natural question is whether DUAL lies in some lower complexity class based on other resources than just runtime. In the present paper, we advance the complexity status of this problem by showing that its complement is feasible with *limited nondeterminism*, i.e, by a nondeterministic polynomial-time algorithm that makes only a poly-logarithmic number of guesses. For a survey on complexity classes with limited nondeterminism, and for several references see [22]. We first show by using a simple but effective technique, which succinctly describes computation paths, that testing non-duality is feasible in polynomial time with $O(\log^3 n)$ nondeterministic steps. We then observe that this approach can be improved to obtain a bound of $O(\chi(n) \cdot \log n)$ nondeterministic steps. *This result is surprising, because most researchers dealing with the complexity of* DUAL *and* TRANS-HYP *believed so far that these problems are completely unrelated to limited nondeterminism.*

We believe that the results presented in this paper are significant, and we are confident that they will be prove useful in various contexts. First, we hope that the various polynomial/output-polynomial cases of the problems which we identify will lead to better and more general methods in various application areas (as we show, e.g. in learning and data mining [11]), and that based on the algorithm DUALIZE or some future modifications, further relevant tractable classes will be identified. Second, we hope that our discovery on limited nondeterminism provides a new momentum to complexity research on DUAL and TRANS-HYP, and will push it towards settling these longstanding open problems.

The rest of this paper is structured as follows. The next section provides some preliminaries and introduces notation. In Section 3, we present our algorithm DUALIZE for dualizing a given monotone prime CNF. In Section 4 we then show that DUAL can be solved with limited nondeterminism.

2 Preliminaries and Notation

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*). 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. 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. We denote by $PI(f)$ the set of all prime implicants of f.

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. We occasionally view CNFs φ also as sets of clauses, and clauses as sets of literals, and use respective notation (e.g., $c \in \varphi$, $\bar{x}_1 \in c$ 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, 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 the unique prime CNF $\varphi = x_2(x_1 \vee x_3)(x_1 \vee x_4)$.

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 a formula for f^d from any one of f by exchanging $\vee$ and $\wedge$ as well as the constants 0 and 1. For example, if f is given by $\varphi = x_1x_2 \vee \bar{x}_1(\bar{x}_3 \vee x_4)$, then f^d is represented by $\psi = (x_1 \vee x_2)(\bar{x}_1 \vee \bar{x}_3x_4)$. For a monotone function f, let $\psi = \bigwedge_{c \in C}(\bigvee_{x_i \in c} x_i)$ be the prime CNF of f^d. Then by De Morgan's law, f has the (unique) prime DNF $\rho = \bigvee_{c \in C}(\bigwedge_{x_i \in c} x_i)$; in the previous example, $\rho = x_1x_2 \vee x_2x_3x_4$. Thus, we will regard DUALIZATION also as the problem of computing the prime DNF of f from the prime CNF of f.

3 Ordered Transversal Generation

In what follows, let f be a monotone function and

$$\varphi = \bigwedge_{i=1}^{m} c_i \qquad (1)$$

its prime CNF, 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 $\varphi_0 = 1$ (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 $\varphi_0 = \varphi_1 = 1$, $\varphi_2 = (x_1 \vee x_2)$, $\varphi_3 = (x_1 \vee x_2)(x_1 \vee x_3)$, and $\varphi_4 = \varphi$. Similarly, for the prime DNF

$$\psi = \bigvee_{t \in PI(f)} t \qquad (2)$$

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.

Proposition 3.1 *Let φ and ψ be any CNF and DNF for f, respectively. Then, for all $i \geq 0$,*

(a) $\|\varphi_i\| \leq \|\varphi\|$ *and* $|\varphi_i| \leq |\varphi|$, *and*

(b) $\|\psi_i\| \leq \|\psi\|$ *and* $|\psi_i| \leq |\psi|$.

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

$$\psi_i \equiv \psi_{i-1} \wedge \Delta^i \equiv \bigvee_{t \in PI(f_{i-1})} (t \wedge \Delta^i). \quad (3)$$

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 (3) that for all $i = 1, 2, \ldots, n$

$$\psi_i \equiv \bigvee_{t \in PI(f_{i-1})} \Big((t \wedge \Delta^i[t]) \vee (t \wedge x_i) \Big). \quad (4)$$

Lemma 3.2 *For every term* $t \in PI(f_{i-1})$, *let* $g_{i,t}$ *be the function represented by* $\Delta^i[t]$. *Then* $|PI(g_{i,t})| \leq |\psi_i| \leq |\psi|$.

We now describe our algorithm DUALIZE for generating $PI(f)$. It is inspired by a similar graph algorithm of Johnson, Yannakakis, and Papadimitriou [30], and can be regarded as a generalization.

Algorithm DUALIZE

Input: The prime CNF φ of a monotone function f.

Output: The prime DNF ψ of f, i.e., all prime implicants of f.

Step 1: Compute the smallest prime implicant t_{min} of f and set $Q := \{ t_{min} \}$;

Step 2: **while** $Q \neq \emptyset$ **do**
Remove the smallest t from Q and output t;
for each i with $x_i \in V(t)$ and $\Delta^i[t] \neq 1$ **do**
Compute the prime DNF $\rho_{(t,i)}$ of the function represented by $\Delta^i[t]$;
for each term t' in $\rho_{(t,i)}$ **do**
if $t_{i-1} \wedge t'$ is a prime implicant of f_i
then Compute the smallest prime implicant t^* of f such that $t_i^* = t_{i-1} \wedge t'$;
$Q := Q \cup \{t^*\}$
end{if} **end**{for} **end**{for}
end{while}

Here, we say that term s is *smaller* than term t if $\sum_{x_j \in V(s)} 2^{n-j} < \sum_{x_j \in V(t)} 2^{n-j}$; i.e., as vector, s is lexicographically smaller than t.

Theorem 3.3 *Algorithm* DUALIZE *correctly outputs all* $t \in PI(f)$ *in increasing order.*

Remark 3.1 (1) The decomposition rule (4) was already used in [32].
(2) In step 1, we could generate any prime implicant t of f, and choose then a lexicographic term ordering inherited from a dynamically generated variable ordering. In step 2, it is sufficient that any monotone DNF $\tau_{(t,i)}$ of the function represented by $\Delta^i[t]$ is computed, rather than its prime DNF $\rho_{(t,i)}$. This might make the algorithm faster.

Let us consider the time complexity of algorithm DUALIZE. We store Q as a binary tree, where each leaf represents a term t and the left (resp., right) son of a node at depth $j - 1 \geq 0$, where the root has depth 0, encodes $x_j \in V(t)$ (resp., $x_j \notin V(t)$). In Step 1, we can compute t_{min} in $O(\|\varphi\|)$ time and initialize Q in $O(n)$ time.

As for Step 2, let $T_{(t,i)}$ be the time required to compute the prime DNF $\rho_{(t,i)}$ from $\Delta^i[t]$. By analyzing its substeps, we can see that each iteration of Step 2 requires $\sum_{x_i \in V(t)} (T_{(t,i)} + |\rho_{(t,i)}| \cdot O(\|\varphi\|))$ time.

Indeed, we can update Q (i.e., remove the smallest term and add t^*) in $O(n)$ time. For each t and i, we can construct $\Delta^i[t]$ in $O(\|\varphi\|)$ time. Moreover, we can check whether $t_{i-1} \wedge t'$ is a prime implicant of f_i and if so, we can compute the smallest prime implicant t^* of f such that $t_i^* = t_{i-1} \wedge t'$ in $O(\|\varphi\|)$ time; note that t^* is the smallest prime implicant of the function obtained from f by fixing $x_j = 1$ if $x_j \in V(t_i \wedge t')$ and 0 if $x_j \notin V(t_i \wedge t')$ for $j \leq i$.

Hence, we have the following result.

Theorem 3.4 *The output delay of Algorithm* DUALIZE *is bounded by*

$$\max_{t \in PI(f)} \Big(\sum_{x_i \in V(t)} (T_{(t,i)} + |\rho_{(t,i)}| \cdot O(\|\varphi\|)) \Big) \quad (5)$$

time, and DUALIZE *needs in total time*

$$\sum_{t \in PI(f)} \sum_{x_i \in V(t)} (T_{(t,i)} + |\rho_{(t,i)}| \cdot O(\|\varphi\|)). \quad (6)$$

Corollary 3.5 *Let* $T = \max\{ T_{(t,i)} \mid t \in PI(f), x_i \in V(t) \}$. *Then, if* T *is bounded by a*

(i) polynomial in n *and* $\|\varphi\|$, *then* DUALIZE *is an* $O(n\|\varphi\|T)$ *polynomial delay algorithm;*

(ii) polynomial in n, $\|\varphi\|$, *and* $\|\psi\|$, *then* DUALIZE *is an* $O(n\cdot|\psi|\cdot(T+|\psi|\cdot\|\varphi\|))$ *polynomial total-time algorithm; moreover,* DUALIZATION *is solvable in incremental polynomial time.*

Note that using results from [2], we can construct from DUALIZE an incremental polynomial time algorithm for DUALIZATION, which however might not output $PI(f)$ in increasing order.

Although we skip the details due to the space limitation, we identify sufficient conditions for the boundedness of T and fruitfully apply them to solve open problems and improve previous results (see Introduction).

4 Limited Nondeterminism

In the previous section, we have discussed polynomial cases of monotone dualization. In this section, we now turn to the issue of the precise complexity of this problem. For this purpose, we consider the decision problem DUAL, i.e., decide whether given monotone prime CNFs φ and ψ represent dual Boolean functions, instead of the search problem DUALIZATION.

It appears that problem DUAL can be solved with limited nondeterminism, i.e., with poly-log many guessed bits by a polynomial-time non-deterministic Turing machine. This result might bring new insight towards settling the complexity of the problem.

We adopt Kintala and Fischer's terminology [31] and write $g(n)$-P for the class of sets accepted by a nondeterministic Turing machine in polynomial time making at most $g(n)$ nondeterministic steps on every input of length n. For every integer $k \geq 1$, define $\beta_k\mathrm{P} = \bigcup_c (c\log^k n)$-P. The $\beta\mathrm{P}$ *Hierarchy* consists of the classes

$$\mathrm{P} = \beta_1\mathrm{P} \subseteq \beta_2\mathrm{P} \subseteq \cdots \subseteq \bigcup_k \beta_k\mathrm{P} = \beta\mathrm{P}$$

and lies between P and NP. The β_kP classes appear to be rather robust; they are closed under polynomial time and logspace many-one reductions and have complete problems (cf. [22]). The complement class of β_kP is denoted by co-β_kP.

Theorem 4.1 *Deciding whether monotone CNFs* φ *and* ψ *are non-dual is feasible in polynomial time with* $O(\chi(n)\log n)$ *nondeterministic steps, where* $n = |\varphi| + |\psi|$.

While our independently developed methods are different from those in [1], the previous result is also obtained from Beigel and Fu's Theorem 11 in [1] by combining [16]. They show how to convert certain recursive algorithms that use disjunctive self-reductions and have runtime bounded by $f(n)$ into polynomial algorithms using $\log f(n)$ nondeterministic steps (cf. [1, Chapter 5]). However, this yields a somewhat more complicated nondeterministic algorithm.

5 Conclusion

We have presented several new cases of the monotone dualization problem which are solvable in output-polynomial time. These cases generalize some previously known output-polynomial cases. Furthermore, we have shown by rather simple means that non-dual monotone pairs (φ, ψ) can be recognized, using a nondeterministic variant of Fredman and Khachiyan's algorithm B [16], in polynomial time with $O(\log^2 n)$ many bit guesses, which places problem DUAL in the class co-β_2P. In fact, a refined analysis revealed that this is feasible in polynomial time with $O(\chi(n)\cdot\log n)$ many bit guesses.

While our results document progress on DUAL and DUALIZATION and reveal novel properties of these problems, the question whether dualization of monotone pairs (φ, ψ) is feasible in polynomial time remains open. It would be interesting to see whether the amount of guessed bits can be further significally decreased, e.g., to $O(\log\log v \cdot \log v)$ many bits.

References

[1] R. Beigel and B. Fu. Molecular computing, bounded nondeterminism, and efficient recursion. *Algorithmica*, 25: 222–238, 1999.

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

[3] 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(4):253–266, 2000.

[4] E. Boros, K. Elbassioni, V. Gurvich, L. Khachiyan and K. Makino. On generating all minimal integer solutions for a monotone system of linear inequalities. In: *Proc. 28th International Colloquium on Automata, Languages and Programming (ICALP)*, pp. 22–103, Springer LNCS 2076, 2001.

[5] E. Boros, V. Gurvich, L. Khachiyan and K. Makino. Dual-bounded generating problems: Partial and multiple transversals of a Hypergraph. *SIAM Journal on Computing*, 30:2036–2050, 2001.

[6] E. Boros, V. Gurvich, L. Khachiyan and K. Makino. On the complexity of generating maximal frequent and minimal infrequent sets. In: *Proc. 19th International Symposium on retical Aspects of Computer Science (STACS)*, pp. 133–141, Springer LNCS 2285, 2002.

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

[8] E. Boros, P. L. Hammer, T. Ibaraki and K. Kawakami, Polynomial time recognition of 2-monotonic positive Boolean functions given by an oracle, *SIAM Journal on Computing*, 26 (1997) 93-109.

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

[10] C. Domingo. Private communication.

[11] 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.

[12] C. Chekuri and A. Rajaraman. Conjunctive query containment revisited. In: *Proc. 6th International Conference on Database Theory (ICDT)*, Delphi, Greece, Springer LNCS 1186, pp. 56–70, 1997.

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

[14] 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.

[15] R. Fagin. Degrees of acyclicity for hypergraphs and relational database schemes. *Journal of the ACM*, 30:514–550, 1983.

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

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

[18] D.R. Gaur. *Satisfiability and self-duality of monotone Boolean functions*. Ph.D. thesis, School of Computing Science, Simon Fraser University, January 1999.

[19] D.R. Gaur and R. Krishnamurti. Self-duality of bounded monotone Boolean functions and related problems. In: *Proc. 11th International Conference on Algorithmic Learning Theory (ALT)*, pp. 209-223, Springer LNCS 1968, 2000.

[20] G. Gogic, C. Papadimitriou, and M. Sideri. Incremental recompilation of knowledge. *Journal of Artificial Intelligence Research*, 8:23–37, 1998.

[21] D. Gunopulos, R. Khardon, H. Mannila, and H. Toivonen. Data mining, hypergraph transversals, and machine learning. *Proc. 16th ACM Symp. on Principles of Database Systems (PODS)*, pp. 209–216, 1997.

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

[23] G. Gottlob, N. Leone, and F. Scarcello. Hypertree decompositions and tractable queries. In: *Proc. 18th ACM Symp. on Principles of Database Systems (PODS)*, pp. 21-32, 1999. Full paper to appear in *Journal of Computer and System Sciences*.

[24] M. Graham. On the universal relation. Technical Report, University of Toronto, Canada, September 1979.

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

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

[27] D. Kavvadias, C. H. Papadimitriou, and M. Sideri, On Horn envelopes and hypergraph transversals. In: *Proc. 4th International Symposium on Algorithms and Computation (ISAAC)*, pp. 399–405, Springer LNCS 762, 1993.

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

[29] 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, June 20-27, 1991.

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

[31] C.M.R. Kintala and P. Fischer. Refining nondeterminism in relativized polynomial-time bounded computations. *SIAM Journal on Computing*, 9:46–53, 1980.

[32] 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.

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

[34] K. Makino and T. Ibaraki. The maximum latency and identification of positive Boolean functions. *SIAM Journal on Computing*, 26:1363–1383, 1997.

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

[36] K. Makino. Efficient dualization of $O(\log n)$-term monotone disjunctive normal forms. Technical Report 00-07, Discrete Mathematics and Systems Science, Osaka University, 2000; to appear in *Discrete Applied Mathematics*.

[37] H. Mannila and K.-J. Räihä. Design by example: An application of Armstrong relations. *Journal of Computer and System Sciences*, 22(2):126–141, 1986.

[38] N. Mishra and L. Pitt. Generating all maximal independent sets of bounded-degree hypergraphs. In: *Proc. Tenth Annual Conference on Computational Learning Theory (COLT)*, pp. 211–217, 1997.

[39] Ch. H. Papadimitriou. NP-completeness: A retrospective, In: *Proc. 24th International Colloquium on Automata, Languages and Programming (ICALP)*, pp.2–6, Springer LNCS 1256, 1997.

[40] U. N. 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.

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

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

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

[44] N. Robertson and P. Seymour. Graph minors II: Algorithmic aspects of tree-width. *Journal of Algorithms*, 7:309–322, 1986.

[45] 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.

[46] H. Tamaki. Space-efficient enumeration of minimal transversals of a hypergraph. IPSJ-AL 75:29-36, 2000.

[47] R. E. Tarjan and M. Yannakakis. Simple linear time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM Journal on Computing*, 13:566–579, 1984.

[48] V. D. Thi. Minimal keys and antikeys. *Acta Cybernetica*, 7(4):361–371, 1986.

[49] B. Toft. Colouring, Stable sets and perfect graphs. *Handbook of Combinatorics*, Vol. 1 Chapter 4. Elsevier, 1995.

[50] C. T. Yu and M. Ozsoyoglu. An algorithm for tree-query membership of a distributed query. *Proceedings IEEE COMPSAC*, pp. 306–312, 1979.