INFORMATION AND COMPUTATION 123, 50–63 (1995)

Complexity of Identification and Dualization of Positive Boolean Functions

JAN C. BIOCH

Department of Computer Science, AI-Section, Erasmus University Rotterdam, P.O. Box 1738, 3000 DR Rotterdam, The Netherlands
E-mail: bioch@cs.few.eur.nl

AND

TOSHIHIDE IBARAKI

Department of Applied Mathematics and Physics, Faculty of Engineering, Kyoto University, Kyoto, Japan 606
E-mail: ibaraki@kuamp.kyoto-u.ac.jp

We consider in this paper the problem of identifying min $T(f)$ and max $F(f)$ of a positive (i.e., monotone) Boolean function f, by using membership queries only, where min $T(f)$ (max $F(f)$) denotes the set of minimal true vectors (maximal false vectors) of f. It is shown that the existence of an incrementally polynomial algorithm for this problem is equivalent to the existence of the following algorithms, where f and g are positive Boolean functions:

(i) An incrementally polynomial algorithm to dualize f;

(ii) An incrementally polynomial algorithm to self-dualize f;

(iii) A polynomial algorithm to decide if f and g are mutually dual;

(iv) A polynomial algorithm to decide if f is self-dual;

(v) A polynomial algorithm to decide if f is saturated;

(vi) A polynomial algorithm in $|\min T(f)| + |\max F(f)|$ to identify min $T(f)$ only.

Some of these are already well known open problems in the respective fields. Other related topics, including various equivalent problems encountered in hypergraph theory and theory of coteries (used in distributed systems), are also discussed.

1. INTRODUCTION

In this paper, we are interested in the problem of identifying a given Boolean function (or a function in short) f by asking membership queries to an oracle whether $f(u) = 0$ or 1 holds for some selected vectors u. In the terminology of computational learning theory [1, 3, 29], this is nothing but the exact learning of a Boolean theory f by membership queries only. It is also a process of forming a theory that explains a certain phenomenon by collecting positive and negative data (in the sense of causing and not causing that phenomenon) [11]. In this case, each membership query is interpreted as conducting an experiment with a given specification to observe its outcome.

We emphasize here that, by identification, we mean to acquire explicit knowledge of *both* $T(f)$ (set of true vectors of f) and $F(f)$ (set of false vectors of f). Although this may appear to be redundant, since $F(f)$ can be computed from $T(f)$ in principle, we consider obtaining both $T(f)$ and $F(f)$ to be essential. During the identification process, which we shall consider in this paper, only partial knowledge of $T(f)$ and $F(f)$ is available. In order to ensure that this partial knowledge actually has become the full knowledge, it is necessary to know both $T(f)$ and $F(f)$. In this sense, maintaining both $T(f)$ and $F(f)$ is natural.

We also note in this regard that the complexity of computing $F(f)$ from $T(f)$ is in general nontrivial. As an example, consider that a function f_G is defined for a graph $G = (V, E)$ by

$$f_G(x) = 1 \text{ if } x \geqslant x(e) \text{ for some } e \in E, \text{ and 0 otherwise,}$$

where $x(e)$ for $e = (v_i, v_j) \in E$ is the vector with $x_i = x_j = 1$ and $x_k = 0$ for $k \neq i, j$. In other words, f_G is a representation of graph G. Now it is not difficult to see that $x \in F(f_G)$ if and only if the set of vertices $\{v_i \in V \mid x_i = 1\}$ is an independent set of G. Computing all maximal independent sets of a graph G is not a trivial task, though it is known that it can be done in incrementally polynomial time (e.g., [19, 22]).

If we do not assume any property for the function f to be identified, identification is not possible unless all values of $f(v)$ for $v \in \{0, 1\}^n$ are explicitly tested. To avoid this situation, we assume in this paper that f is known to be *positive* (i.e., *monotone*), since this is the case in many applications. (The definition of a positive functions and related concepts will be given in subsequent sections.) For example, if $f(x)$ describes the result of diagnosing a disease from the given data (i.e., symptoms) x, e.g., x_1 denotes whether temperature is high ($x_1 = 1$) or not ($x_1 = 0$), and x_2 denotes whether blood presure is high ($x_1 = 1$) or not ($x_1 = 0$), etc., it would be natural to assume that we somehow know the direction of each variable that tends to cause the disease to

0890-5401/95 $12.00

appear. In this case, by adjusting the polarities of variables if necessary, the function $f(x)$ can be assumed to be positive without loss of generality.

If f is a positive function, $T(f)$ and $F(f)$ can be compactly represented by min $T(f)$ (set of minimal true vectors) and max $F(f)$ (set of maximal false vectors). Therefore we want to compute min $T(f)$ and max $F(f)$ instead of $T(f)$ and $F(f)$ themselves in our identification process. The complexity of this type of enumeration algorithm is usually measured in its lengths of input and output. A particularly interesting class from this view point is that of incrementally polynomial algorithms [19]. An algorithm to enumerate items $a_1, a_2, \ldots, a_p$ is called *incrementally polynomial* (i) if it iterates the following procedure for $i = 1, 2, \ldots, p$: output the ith item a_i from the knowledge of its input and items $a_1, a_2, \ldots, a_{i-1}$ and (ii) if the time required for the ith iteration is polynomial in the input length and the sum of the sizes of $a_1, a_2, \ldots, a_{i-1}$.

In our problem of identification, the input is considered to be an oracle of f, which answers the membership queries given to it. We define its input length to be n (the number of variables) since the set of variables on which f is defined must be explicitly specified. Given a membership query (i.e., a vector $u \in \{0, 1\}^n$), the oracle returns its answer (i.e., $f(u)$) in one unit of time.

Our main question is to know whether the identification of a positive f has an incrementally polynomial algorithm or not. Unfortunately, we could not resolve this question in this paper. However, we could relate the existence of such an algorithm with many other interesting problems, discussed not only in Boolean theory but also in different fields such as hypergraph theory [12], theory of coteries (used in distributed systems) [15, 18, 21], database theory [12, 24] and artificial intelligence [12, 20, 27]. In particular, we show that an incrementally polynomial identification algorithm exists if and only if dualization of a positive function has an incrementally polynomial algorithm. Also such algorithms exist if and only if problems such as deciding if two positive functions are mutually dual, deciding if a positive function is self-dual, deciding if a simple hypergraph is saturated and deciding if a coterie is nondominated, can be solved in polynomial time. All these problems are polynomially equivalent, but their complexity is still left open. It is again noted here that our problem is different from that of identifying only min $T(f)$ or only max $F(f)$ since it is known [1, 22] that no incrementally polynomial algorithm exists for this problem.

It can be shown that the polynomial equivalence among the problems stated above carries over to a subclass of positive functions if the class satisfies some additional conditions. For several special classes of positive functions, it is known that some problems (e.g., dualization) in the above list can be solved by incrementally polynomial algorithms. Such classes include (i) the class of 2-monotonic positive functions [5, 9, 10, 26], which include as a special case the class of positive threshold functions [25], (ii) the class of positive k-DNF (disjunctive normal form) [12], where a positive function f is k-DNF if each minimal true vector has at most k elements of 1, and (iii) other classes (e.g., [12, 23]). Based on these, it is possible in particular to show that positive functions in all these special classes can be identified by incrementally polynomial algorithms. Another special class, called read-once functions, also deserve mentioning here, since it is shown in [2] that this class has an algorithm to identify its read-once formula (i.e., an expression of $T(f)$) in polynomial time in its formula length (note, however, that the framework for identification is different from that of this paper). Finally, although not discussed in this paper, other classes different from that of positive functions, such as general threshold functions, Horn functions, and general k-DNF functions, may be fruitful to study.

Some other results related to our main question are also included in this paper. It turns out that some relevant problems are solvable in polynomial time, while others are NP-complete. This may indicate that our problem is located somewhere close to the border between polynomiality and NP-hardness.

In concluding this section, we should add a recent important result by Fredman and Khachiyan [13] showing that the mutual duality between two positive functions f and g can be tested in $O(m^{o(\log m)})$ time, where m is the number of prime implicants in the DNFs of f and g. This tells that the problem of checking mutual duality (and hence all the equivalent problems discussed in this paper) is unlikely to be NP-hard. The above complexity is not polynomial, however, and it is still open whether polynomial time algorithms really exist or not.

2. IDENTIFICATION OF POSITIVE BOOLEAN FUNCTIONS

2.1. *Definitions and an Algorithm*

A *Boolean function*, or a *function* in short, is a mapping $f: \{0, 1\}^n \mapsto \{0, 1\}$, where $v \in \{0, 1\}^n$ is called a *Boolean vector* (a *vector* in short). If $f(v) = 1$ (resp. 0), then v is called a *true* (resp. *false*) vector of f. The set of all true vectors (false vectors) is denoted by $T(f)$ ($F(f)$). Two special functions with $T(f) = \varnothing$ and $F(f) = \varnothing$ are respectively denoted by $f(x) = \bot$ and $f(x) = \top$.

A function f is *positive* if $v \leqslant w$ always implies $f(v) \leqslant f(w)$. A positive function is also called *monotone*. (However, we do not use the latter name since some people use it to imply a function that is either monotone nonincreasing or monotone nondecreasing.) A true vector v of f is *minimal* if there is no other true vector w such that $w < v$, and let min $T(f)$ denote the set of all minimal true vectors of f. A *maximal* false vector is symmetrically defined and max $F(f)$

denotes the set of all maximal false vectors of f. Two vectors u and v are *incomparable* if neither $u \geq v$ nor $v \geq u$ holds. A set of vectors $U \subseteq \{0,1\}^n$ is called *incomparable* if every pair of vectors $u, v \in U$ is incomparable. It is clear that the sets $\min T(f)$ and $\max F(f)$ are both incomparable.

If f is positive, it is known [25, 30] that f has the unique disjunctive normal form (DNF) consisting of all prime implicants. There is one-to-one correspondence between prime implicants and minimal true vectors. For example, a positive function $f = 12 \vee 23 \vee 31$, where 12 stands for x_1x_2 and so on, has prime implicants 12, 23, 31 which correspond to minimal true vectors (110), (011), (101), respectively. In other words, the input length to describe a positive function f is $O(n\,|\min T(f)|)$ if it is represented in this manner. Furthermore, the sets $\min T(f)$ and $\max F(f)$ conversely define $T(f)$ and $F(f)$ by

$$T(f) = \{v \mid v \geq w \text{ for some } w \in \min T(f)\}$$
$$F(f) = \{v \mid v \leq w \text{ for some } w \in \max F(f)\}.$$

Since $|\min T(f)| \leq |T(f)|$ and $|\max F(f)| \leq |F(f)|$, we ask in our identification algorithm to compute $\min T(f)$ and $\max F(f)$ instead of $T(f)$ and $F(f)$. Thus our problem is stated as follows, where the input length of an oracle for f is considered to be n, as mentioned in the Introduction.

Problem IDENTIFICATION.

Input: An oracle for a positive function f.

Output: $\min T(f)$ and $\max F(f)$.

Now let MT and MF respectively denote the partial knowledge of $\min T(f)$ and $\max F(f)$ currently at hand, i.e.,

$$MT \subseteq \min T(f) \quad \text{and} \quad MF \subseteq \max F(f). \tag{1}$$

Define

$$T(MT) = \{v \mid v \geq w \text{ for some } w \in MT\}$$
$$F(MF) = \{v \mid v \leq w \text{ for some } w \in MF\}. \tag{2}$$

By assumption (1), $T(MT) \subseteq T(f)$ and $F(MF) \subseteq F(f)$, and hence

$$T(MT) \cap F(MF) = \varnothing \tag{3}$$

holds. A vector u is called an *unknown* vector if

$$u \in \{0,1\}^n \backslash (T(MT) \cup F(MF)), \tag{4}$$

since it is not known at the current stage whether u is a true vector or a false vector of f. There is no unknown vector if and only if $T(MT) \cup F(MF) = \{0,1\}^n$ holds; i.e., $MT = \min T(f)$ and $MF = \max F(f)$ hold for some positive function f.

The general procedure of identifying a positive function f can now be described as follows.

ALGORITHM IDENTIFY.

Input: An oracle for a positive function f.

Output: $\min T(f)$ and $\max F(f)$.

1. Start with appropriate sets $MT (\subseteq \min T(f))$ and $MF (\subseteq \max F(f))$ (e.g., $MT := MF := \varnothing$).

2. Test if $T(MT) \cup F(MF) = \{0,1\}^n$ holds. If so, output MT and MF, and halt. Otherwise go to 3.

3. Find an unknown vector u, and ask an oracle if $f(u) = 1$ or $f(u) = 0$. If $f(u) = 1$, then compute a new minimal true vector y from u and let $MT := MT \cup \{y\}$. On the other hand, if $f(u) = 0$, compute a new maximal false vector y from u and let $MF := MF \cup \{y\}$. Return to 2.

This IDENTIFY is more or less a standard algorithm discussed in the framework of learning theory. Among rich variety of enumeration algorithms in discrete mathematics, there are also many algorithms, such as those used in [19, 22] and others to enumerate all maximal independent sets, which have certain resemblance to IDENTIFY (though there are nontrivial differences in details).

2.2. *Analysis of Algorithm* IDENTIFY

We are particularly interested in whether Algorithm IDENTIFY can be incrementally polynomial or not. For this, Step 2 is crucial, and we define:

Problem EQ.

Input: Incomparable sets MT, $MF (\subseteq \{0,1\}^n)$ such that $T(MT) \cap F(MF) = \varnothing$.

Question: Does $T(MT) \cup F(MF) = \{0,1\}^n$ (i.e., no unknown vector) hold?

Unfortunately the complexity of Problem EQ is not known yet, but will be related to other well known problems, in Section 3. We show below, however, that if Problem EQ (which is a decision problem) is solved in time polynomial in its input length $n(|MT| + |MF|)$, then finding an unknown vector in Step 3 can also be done in polynomial time.

For this, introduce the following notations. Let $a = (a_1, a_2, \ldots, a_k) \in \{0,1\}^k$ for some $k < n$, and define

$$\begin{aligned}
MT[a] &= \text{MinSet}\{(v_{k+1}, \ldots, v_n) \mid v \in MT \\
&\qquad \text{and } v_i \leq a_i,\ i = 1, 2, \ldots, k\} \\
\min T(f)[a] &= \text{MinSet}\{(v_{k+1}, \ldots, v_n) \mid v \in \min T(f) \\
&\qquad \text{and } v_i \leq a_i,\ i = 1, 2, \ldots, k\} \\
MF[a] &= \text{MaxSet}\{(v_{k+1}, \ldots, v_n) \mid v \in MF \\
&\qquad \text{and } v_i \geq a_i,\ i = 1, 2, \ldots, k\} \\
\max F(f)[a] &= \text{MaxSet}\{(v_{k+1}, \ldots, v_n) \mid v \in \max F(f) \\
&\qquad \text{and } v_i \geq a_i,\ i = 1, 2, \ldots, k\},
\end{aligned} \tag{5}$$

where MinSet(A) (MaxSet(A)) denotes the set of minimal (maximal) vectors in A. Denote by f_a the function obtained from f by fixing variables x_i to a_i for $i = 1, 2, ..., k$. Then the above min $T(f)[a]$ and max $F(f)[a]$ represent min $T(f_a)$ and max $F(f_a)$, respectively.

Now, as noted previously, $MT = \min T(f)$ and $MF = \max F(f)$ hold if and only if $T(MT) \cup F(MF) = \{0, 1\}^n$. Furthermore, if $MT \subseteq \min T(f)$ and $MF \subseteq \max F(f)$, then $MT[a] \subseteq \min T(f)[a]$ and $MF[a] \subseteq \max F(f)[a]$ for any a. Also $MT[a] = \min T(f)[a]$ and $MF[a] = \max F(f)[a]$ hold if and only if

$$T(MT[a]) \cup F(MF[a]) = \{0, 1\}^{n-k}. \tag{6}$$

If the current $MT[a]$ and $MF[a]$ do not satisfy condition (6), then it follows that at least one of $a^0 = (a, 0)$ and $a^1 = (a, 1)$ does not satisfy condition (6) (in which a is replaced by a^0 and a^1, respectively), since

$$\begin{aligned} T(MT[a]) &= T(\{(0, w) \mid w \in MT[a^0]\}) \\ &\quad \cup T(\{1, w) \mid w \in MT[a^1]\}) \\ F(MF[a]) &= F(\{(0, w) \mid w \in MF[a^0]\}) \\ &\quad \cup F(\{(1, w) \mid w \in MF[a^1]\}), \end{aligned}$$

by definition.

Based on this property, the following algorithm finds an unknown vector u by systematically searching vectors a that do not satisfy the above condition (6), by increasing the dimension of a at each iteration.

ALGORITHM UNKNOWN.

Input: Incomparable sets MT, $MF (\subseteq \{0, 1\}^n)$ such that $T(MT) \cap F(MF) = \varnothing$ and $T(MT) \cup F(MF) \neq \{0, 1\}^n$, where $n \geq 2$.

Output: An unknown vector u.

1. Let $a^0 := (0)$, $a^1 := (1)$ and $k := 1$. Go to 2.

2. If $k = n - 1$, then at least one of $M_0 = \{0, 1\} \setminus (MT[a^0] \cup MF[a^0])$ and $M_1 = \{0, 1\} \setminus (MT[a^1] \cup MF[a^1])$ is nonempty. If $M_0 \neq \varnothing$ and $b \in M_0$, let $u := (a^0, b)$ and halt. Otherwise, let $u := (a^1, b)$, where $b \in M_1$, and halt. If $k < n - 1$, go to 3.

3. Test if $T(MT[a^0]) \cup F(MF[a^0]) = \{0, 1\}^{n-k}$ holds (i.e., solve Problem EQ). If "no," let $a^0 := (a^0, 0)$, $a^1 := (a^0, 1)$ and $k := k + 1$. Return to 2. Otherwise (i.e., "yes"), let $a^0 := (a^1, 0)$, $a^1 := (a^1, 1)$ and $k := k + 1$. Return to 2.

The running time of UNKNOWN is $O(nT_{EQ}(n(|MT| + |MF|)))$, where $n(|MT| + |MF|)$ is the input length to problem EQ and T_{EQ} is the time to solve Problem EQ, if we ignore the time for MinSet and MaxSet operations of (5) for $MT[a^i]$ and $MF[a^i]$ in Steps 2 and 3, for the time being.

At this point, recall the two kinds of queries, *restricted equivalence query* and *equivalence query*, well studied in computational leaning theory (e.g., [1]). A restricted equivalent query asks whether the current knowledge of f completely characterizes f or not, while an equivalence query asks, in addition to that, to provide a counterexample if it does not characterize f. In our framework, a restricted equivalence query is essentially equal to solving problem EQ, since MT and MF completely characterize f if and only if $T(MT) \cup F(MF) = \{0, 1\}^n$, while an equivalence query is nothing but to solve problem EQ and then to execute Algorithm UNKNOWN, since the latter provides a counterexample when problem EQ has answer "no." The above time complexity of UNKNOWN says that a restricted equivalence query and an equivalence query do not have an essential difference (modulo polynomial time), in our problem setting.

Now we consider the second half of Step 3 of IDENTIFY. Asking an oracle if $f(u) = 0$ or $f(u) = 1$ is called a *membership query*. Computing a minimal true vector or a maximal false vector y from an unknown vector u can be done as described in such papers as [1, 14, 29]. For example, if $f(u) = 0$, a maximal false vector $y (\geq u)$ not in $F(MF)$ can be computed as follows, where $y[y_i = 1]$ denotes the vector y with its ith element y_i fixed to 1.

ALGORITHM MAXIMAL.

Input: An incomparable set $MF \subseteq \max F(f)$, a vector $u \in F(f) \setminus F(MF)$, and an oracle for a positive function f.

Output: A maximal vector y such that $y \in F(f) \setminus F(MF)$.

1. $y := u$.

2. For $i = 1, 2, ..., n$, let $y := y[y_i = 1]$ if $f(y[y_i = 1]) = 0$ (membership queries are used here).

3. Output y.

A minimal true vector can be analogously computed and its algorithm is called MINIMAL. Both algorithms issue $O(n)$ membership queries before outputting the final y. If each membership query in step 2 is answered in constant time by an oracle, the running time of MAXIMAL or MINIMAL is $O(n)$.

The sum of the above running times in IDENTIFY becomes

$$\begin{aligned} &O((|\min T(f)| + |\max F(f)|)\, nT_{EQ}(n(|MT| + |MF|))) \\ &\quad = O(mnT_{EQ}(mn)), \end{aligned} \tag{7}$$

where m is defined by

$$m = |\min T(f)| + |\max F(f)|. \tag{8}$$

Before proceeding further, recall that we have postponed the discussion on the time complexity of MinSet and

MaxSet operations of (5) required in Steps 2 and 3 of UNKNOWN. For simplicity, we explain how to facilitate this computation only for MT, since MF can be analogously treated. Prepare data PAIR(MT, k), $k = 1, 2, ..., n$, defined by $(u, v) \in$ PAIR(MT, k) if (a) $u, v \in MT$, (b) $(u_{k+1}, ..., u_n) < (v_{k+1}, ..., v_n)$, or $(u_{k+1}, ..., u_n) = (v_{k+1}, ..., v_n)$ and $Num(u) < Num(v)$, where u is the $Num(u)$th vector stored in the array of MT, and (c) $(u_{i+1}, ..., u_n)$ and $(v_{i+1}, ..., v_n)$ are incomparable for all $i < k$. PAIR(MT, k) for all $k = 1, 2, ..., n$ can be constructed in $O(n\,|MT|^2)$ ($\leqslant O(nm^2)$) time. Using this data, MinSet operation to obtain $MT[a^0]$ from $MT[a]$ can be carried out as follows, where $a^0 = (a, 0)$ and the dimension of a^0 is k. (The case of $a^1 = (a, 1)$ is analogous.) First obtain $MT' = \{v \mid v \in MT[a],\ v_k = 0\}$ from $MT[a]$ in $O(m)$ time. Then $v \in MT[a^0]$ holds if and only if $v \in MT'$ and there is no $u \in MT'$ such that $(u, v) \in$ PAIR(MT, k). If we use a data structure of MT', with which $v \in MT'$ can be checked in constant time, then $MT[a^0]$ can be obtained from MT' in $O(n\,|\text{PAIR}(MT, k)|)$ time. The overall time for all k is $O(mn)$ for constructing MT', and $\sum_k O(n\,|\text{PAIR}(MT, k)|) = O(n\,|MT|^2)$ for obtaining $MT[a^0]$ and $MT[a^1]$. Therefore, one execution of UNKNOWN can be carried out in $O(nm^2))$ time.

Since UNKNOWN is called at most m times in IDENTIFY, the overall time needed for MinSet and MaxSet operations is $O(nm^3))$. This time bound is dominated by the above time bound (7) if we assume $T_{EQ}(nm) \geqslant O(m^2)$. For this reason, we shall not consider explicitly the time for MinSet and MaxSet operations in the subsequent discussion.

As a summary of the above discussion, we have the next theorem.

THEOREM 2.1. *Problem* IDENTIFICATION *has an incrementally polynomial algorithm if and only if problem EQ can be solved in polynomial time.*

Proof. The if-part is immediate from (7). To prove the converse, execute the incrementally polynomial algorithm A for Problem IDENTIFICATION (note that A may be quite different from IDENTIFY), in which a positive function f is given by min $T(f) = MT$, until either (i) it halts or (ii) it iterates $|MT| + |MF| + 1$ times. In case (i), there is no unknown vector if and only if the obtained max $F(f)$ satisfies max $F(f) = MF$. In case (ii), there is an unknown vector since $|\max F(f)| > |MF|$ holds. In any case, Problem EQ is solved. Since a membership query for u can be answered in time $O(n\,|MT|)$ by directly checking the property: $f(u) = 1 \Leftrightarrow u \geqslant v$ for some $v \in MT$, without resorting to an oracle, the time to reach (i) or (ii) is $O((|MT| + |MF|)\,p(n(|MT| + |MF|))\,(n\,|MT|))$, where $p(\cdot)$ is the time for one iteration of Algorithm A. The third factor $n\,|MT|$ takes into account the time to answer membership queries (which is constant if an oracle is used). The time bound is polynomial in the input length $n(|MT| + |MF|)$ of problem EQ. ∎

The existence of an incrementally polynomial algorithm for a problem implies that the problem can be solved in *polynomial total time* [19], i.e., polynomial time in the length of input and output. Although the converse is not generally true, we show below that it is true for Problem IDENTIFICATION.

COROLLARY 2.1. *Problem* IDENTIFICATION *can be solved in polynomial total time if and only if it has an incrementally polynomial algorithm.*

Proof. The if-part is immediate from definition. The only-if-part is proved by showing that EQ can be solved in polynomial time if IDENTIFICATION can be solved by an Algorithm B that runs in polynomial total time $p(n, m)$, where $m = n(|\min T(f)| + |\max F(f)|)$ is the output length. (The rest follows immediately from Theorem 2.1.) Now, given an instance of EQ with MT and MF, satisfying (1), execute Algorithm B only $p(n, m')$ time, where $m' = n(|MT| + |MF|)$ is the input length of EQ. If B halts within this time bound, EQ has answer "yes" if and only if min $T(f)$ and max $F(f)$ computed by B satisfy $MT = \min T(f)$ and $MF = \max F(f)$. On the other hand, if B does not halt within this time bound, EQ has answer "no," since it implies $|\min T(f)| + |\max F(f)| > |MT| + |MF|$. In any case, EQ is solved in $p(n, m')$ time, which is polynomial in its input length m'. ∎

3. DUAL FUNCTIONS AND RELATED PROBLEMS

3.1. *Definitions*

The *dual* of a Boolean function f, denoted f^d, is defined by $f^d(x) = \bar{f}(\bar{x})$, where $\bar{f}$ and $\bar{x}$ denote the complements of f and x, respectively. As is well known, the DNF expression of f^d is obtained from that of f by exchanging $\vee$ (or) and $\cdot$ (and), as well as constants 0 and 1, and then expanding the resulting formula. (The symbol $\cdot$ of "and" is usually omitted unless confusion occurs.) For example, the dual of $f = 12 \vee 23 \vee 31$ is $f^d = (1 \vee 2)(2 \vee 3)(3 \vee 1) = 12 \vee 23 \vee 31$. As a special case, functions $f = \bot$ and $f = \top$ are mutually dual. Denote $f \leqslant g$ if these functions satisfy $f(v) \leqslant g(v)$ for all $v \in \{0, 1\}^n$. It is immediate from definition that $f = g^d \Leftrightarrow f^d = g$, and $f \leqslant g \Leftrightarrow f^d \geqslant g^d$. If f is positive, another characterization of f^d is given as follows: $y \in \min T(f^d)$ if and only if y is a minimal vector with property $y \wedge v \neq (00 \cdots 0)$ for all $v \in \min T(f)$, where $y \wedge v = (y_1 \wedge v_1,\ y_2 \wedge v_2, ..., y_n \wedge v_n)$ and $\wedge$ is "and" operator. For this reason, the vectors in $T(f^d)$ ($\min T(f^d)$) are called the (*minimal*) *transversals* of f.

A function f is called *dual-minor* if $f \leqslant f^d$, *dual-major* if $f \geqslant f^d$, and *self-dual* if $f = f^d$. For example, the above function $f = 12 \vee 23 \vee 31$ is self-dual. These definitions have the following characterizations [18]:

(i) f is dual-minor if and only if at most one of v and $\bar{v}$ is true for all $v \in \{0, 1\}^n$.

(ii) f is dual-major if and only if at least one of v and $\bar{v}$ is true for all $v \in \{0, 1\}^n$.

(iii) f is self-dual if and only if exactly one of v and $\bar{v}$ is true for all $v \in \{0, 1\}^n$.

In this paper, we mostly deal with positive functions. As mentioned in Section 2, the length to input a positive function is $O(n\,|\min T(f)|)$. The lengths of f and its dual f^d may be substantially different, however, as exemplified by the following functions [1]:

$$f = x_1y_1 \vee x_2y_2 \vee \cdots \vee x_py_p \quad (\text{length } \Omega(p))$$
$$f^d = \bigvee \{z_1z_2 \cdots z_p \mid z_i \in \{x_i, y_i\}, i = 1, 2, \ldots, p\} \quad (\text{length } \Omega(p2^p)). \tag{9}$$

The definition of the dual, $f^d(x) = \bar{f}(\bar{x})$, implies that

$$T(f^d) = \{\bar{v} \mid v \in F(f)\}, \quad \text{i.e., } F(f) = \{\bar{v} \mid v \in T(f^d)\}, \tag{10}$$

which then implies

$$\min T(f^d) = \{\bar{v} \mid v \in \max F(f)\}, \text{ i.e., } \max F(f) = \{\bar{v} \mid v \in \min T(f^d)\}. \tag{11}$$

Therefore the functions (9) tell that, for a positive function f, the lengths of $|\min T(f)|$ and $|\max F(f)|$ can be very different. It also supports our claim in Section 1 that both $\min T(f)$ and $\max F(f)$ should be explicitly output in the identification process. (Since the positive function f in (9) is 2-DNF, the results in Section 4.4 show that it can be identified in our sense by an incrementally polynomial algorithm.)

Definition (10) of the dual indicates that an oracle of f can also be used as an oracle of f^d. The following is a result of this observation.

COROLLARY 3.1 *Given an oracle for membership queries to a positive function f,* IDENTIFICATION *has an incrementally polynomial algorithm if and only if* $\min T(f)$ *can be computed by an algorithm polynomial in* $|\min T(f)| + |\max F(f)|$.

Proof. The only-if part is obvious. To show the converse, apply the latter algorithm to both f and f^d to compute $\min T(f)$ and $\min T(f^d)$, respectively. Then $\max F(f)$ is obtained by (11). This is a polynomial total time algorithm for IDENTIFICATION. Then apply Corollary 2.1. ∎

The complexity of deciding if a given positive function f is dual-major and/or self-dual will be discussed in the next subsection. Here we note that it can be checked in polynomial time whether f is dual-minor or not.

LEMMA 3.1 [6, 15, 18, 25]. *Let f be a positive function, and assume that f is represented in* DNF.

(i) *f is dual-minor if and only if every pair of prime implicants m_i and m_j of f has at least one literal in common.*

(ii) *The property in* (i) *can be checked in $O(n\,|\min T(f)|^2)$ time.*

3.2. *Complexity of Positive Functions*

We call problems A and B *polynomially equivalent* if A and B are mutually reducible [16]. This tells in particular that, if one of them is solvable in polynomial time, then so is the other. In relation with the notions introduced in the previous subsection, we consider the following problems.

Problem MD.

Input: Positive functions f and g of n variables, i.e., $\min T(f)$ and $\min T(g)$.

Question: Are f and g mutually dual, i.e., $f = g^d$ (hence $g = f^d$)?

Problem SD.

Input: A positive function f of n variables, i.e., $\min T(f)$.

Question: Is f self-dual?

THEOREM 3.1. *Problems* EQ, MD, SD *are polynomially equivalent.*

Proof. (a) MD and SD. SD is a special case of MD, i.e., when $g = f$, and this shows that SD is reducible to MD. To show the converse, given an instance of MD, consider function

$$h = fx \vee gy \vee xy, \tag{12}$$

where none of f and g contain x or y as a variable. Then

$$h^d = (f^d \vee x)(g^d \vee y)(x \vee y) = g^dx \vee f^dy \vee xy, \tag{13}$$

and $h = h^d$ holds (i.e., h is self-dual) if and only if f and g are mutually dual; i.e., $f = g^d$ (hence $f^d = g$).

(b) EQ and MD. Recall property (11). Now, given MT and MF of Problem EQ, define two functions f and g by

$$\min T(f) = MT \quad (\text{i.e., } T(f) = T(MT))$$
$$\min T(g) = \{\bar{v} \mid v \in MF\}.$$

Assumption $T(MT) \cap F(MF) = \varnothing$ in problem EQ implies $F(f) \supseteq F(MF)$; i.e., $f^d \geq g$ (hence $f \leq g^d$). Therefore, $T(MT) \cup F(MF) = \{0, 1\}^n$ holds if and only if $F(f) = F(MF)$; i.e., f and g are mutually dual. This shows that EQ

is reducible to MD. To show the converse reduction, first check if $g^d \geq f$ holds for given positive functions f and g of MD. For this, consider again h and h^d of (12) and (13). It is not difficult to show that $g^d \geq f$ (hence $f^d \geq g$) holds if and only if $h^d \geq h$; i.e., h is dual-minor. As noted in Lemma 3.1, the last condition can be checked in polynomial time. If h is not dual-minor (i.e., $g^d \geq f$ does not hold), then f and g are not mutually dual. Therefore, assume that h is dual-minor. In this case, define

$$MT = \min T(f)$$
$$MF = \{\bar{v} \mid v \in \min T(g)\},$$

where $T(MT) \cap F(MF) = \varnothing$ holds by property $g^d \geq f$. By an argument similar to the above, it is now straightforward to see that $T(MT) \cup F(MF) = \{0, 1\}^n$ holds if and only if f and g are mutually dual. Thus MD is reducible to EQ. ∎

The equivalence between Problems MD and SD was already proved in [12] in hypergraph setting.

The complexity of Problem SD has been one of the important topics in distributed system theory. To explain this, note that a positive function $f_{\mathscr{C}}$ can be used as a convenient tool to represent an incomparable family $\mathscr{C}$ of subsets $S \subseteq \{1, 2, ..., n\}$, by letting $w \in \min T(f_{\mathscr{C}})$ if and only if $w = \chi(S)$ for some $S \in \mathscr{C}$, where $y = \chi(S)$ is the *characteristic vector* of S defined by $y_i = 1$ if $i \in S$ and 0 otherwise. A family $\mathscr{C}$ is said to be *incomparable* if no two $S, S' \in \mathscr{C}$ satisfy $S \subseteq S'$ or $S' \subseteq S$. An incomparable family is also called a 1-*Sperner family* or an *antichain*. Let $\mathscr{C}$ be an incomparable family. It is a *coterie* if it also satisfies the *intersecting property*:

$$S \cap S' \neq \varnothing \quad \text{for all } S, S' \in \mathscr{C}.$$

Coteries have been studied in conjunction with mutual exclusion in a distributed system [15, 18, 21]. A coterie $\mathscr{C}$ is *nondominated* if there is no coterie $\mathscr{C}' \neq \mathscr{C}$ such that, for each $S \in \mathscr{C}$, there is a subset $S' \in \mathscr{C}'$ with $S' \subseteq S$. Nondominated coteries are important in practical applications, and are closely related to Problem SD, as shown in the next lemma.

LEMMA 3.2 [18]. *Let $f_{\mathscr{C}}$ be the positive function representing an incomparable family $\mathscr{C}$ of subsets in $\{1, 2, ..., n\}$. Then,*

(i) *$\mathscr{C}$ is a coterie if and only if $f_{\mathscr{C}}$ is dual-minor, and*

(ii) *$\mathscr{C}$ is a nondominated coterie if and only if $f_{\mathscr{C}}$ is self-dual.*

Further properties of positive self-dual functions are found in [7].

In concluding this subsection, we point out that the following problem, closely related to problem SD, is NP-complete.

Problem NDMAJOR.

Input: A positive function f of n variables, i.e., min $T(f)$.

Question: Is f not dual-major?

THEOREM 3.2. *Problem* NDMAJOR *is* NP-*complete.*

Proof. We reduce the following problem, which is known to be NP-complete [16], to NDMAJOR.

Problem SET SPLITTING.

Input: A family $\mathscr{C}$ of subsets of $V = \{1, 2, ..., n\}$.

Question: Is there a partition of V into two subsets V_1 and V_2 such that no subset in $\mathscr{C}$ is entirely contained in either V_1 or V_2?

First note that we can assume without loss of generality that $\mathscr{C}$ is incomparable. This is because, if $S, S' \in \mathscr{C}$ satisfy $S \subset S'$, then S' can be deleted without changing the result since $S' \subseteq V_i (i = 1 \text{ or } 2)$ implies $S \subseteq V_i$. Then consider the positive function $f_{\mathscr{C}}$, defined after Theorem 3.1. In view of characterization (ii) in Section 3.1, $f_{\mathscr{C}}$ is not dual-major if and only if there is a pair of vectors w and $\bar{w}$ both belonging to $F(f_{\mathscr{C}})$; i.e., there is a partition of V as stated in problem SET SPLITTING (define V_1 and V_2 by $\chi(V_1) = w$ and $\chi(V_2) = \bar{w}$). Since NDMAJOR is obviously in class NP, this proves the lemma. ∎

Essentially the same proofs are also found in [12, 15] and possibly others, though they are stated in different terminologies (i.e., coteries and hypergraphs). If fact, the equivalence between NDMAJOR and SET SPLITTING was already stated in [25].

It may be worth noting at this point that problem SD is asking whether a dual-minor positive function f is at the same time dual-major. The restriction that f is dual-minor may make it easier to decide. However, we do not know yet whether this is in fact the case or not.

3.3. *Complexity of General Functions*

In this section, we consider the problems associated with dual functions of general Boolean functions. In order to deal with general functions, we first discuss how to represent them, since the DNF expression with prime implicants is not unique in this case. Here, we assume that a general function is represented either by a DNF (disjunctive normal form) of implicants (monomials, i.e., conjunctions of literals), or by a CNF (conjunctive normal form) of clauses (i.e., disjunctions of literals). In the former case, we use letters f, g, h and so on, while in the latter case we use ϕ. For example,

$$f = 134 \vee \bar{1}\bar{3}4 \vee \bar{1}23 \vee 123 \vee 1\bar{2}3 \vee \bar{1}3\bar{4}$$
$$= 13 \vee 23 \vee 3\bar{4} \vee \bar{1}\bar{3}4,$$
$$\phi = (1 \vee 3)(2 \vee 3)(3 \vee \bar{4})(\bar{1} \vee \bar{3} \vee 4),$$

where i and $\bar{i}$ stands for x_i and $\bar{x}_i$, respectively.

For the subsequent discussion, we introduce some more definitions. The *contra-dual* f^* of f is defined by $f^*(x) = f(\bar{x})$ [6, 7, 17]. For example, the contra-dual of $f = 12 \vee 23 \vee 31$ is $f^* = \bar{1}\bar{2} \vee \bar{2}\bar{3} \vee \bar{3}\bar{1}$. A function f is called *anti-dual* if $f^* = f$. For example, $f = 1\bar{2} \vee 13 \vee \bar{1}2 \vee \bar{1}\bar{3}$ is anti-dual. Functions such as ff^*, $f^d\bar{f}$ and $f \vee f^*$ are all anti-dual.

LEMMA 3.3. (i) *f is dual-minor if and only if $ff^* = \perp$ (i.e., $T(f) \cap T(f^*) = \varnothing$).*

(ii) *f is anti-dual if and only if there is a dual-minor function g such that $f = g \vee g^*$.*

Proof. (i) Note that $T(f^*) = F(f^d)$ by (10) and the definition of f^*. Therefore, $f \leqslant f^d$ implies $T(f) \cap F(f^d) = \varnothing$, i.e., $T(f) \cap T(f^*) = \varnothing$. The converse is also similar.

(ii) The if-part follows from

$$f^* = \overline{f^d} = \overline{g^d \bar{g}} \text{ (by } (g^*)^d = \bar{g}) = g \vee g^* \text{(by } \overline{g^d} = g^*) = f.$$

To show the converse, represent f by

$$f = fx \vee f\bar{x},$$

for a variable x. Then $g = fx$ is dual-minor since

$$g^d = f^d \vee x \geqslant x \geqslant fx = g,$$

and $g^* = f^*x^* = f\bar{x}$ (by the anti-duality of f). ∎

Now Lemma 3.1 for positive functions can be extended to general functions.

THEOREM 3.3. (i) *Let a function f be given by*

$$f = m_1 \vee m_2 \vee \cdots \vee m_p,$$

where m_i are implicants of f. Then f is dual-minor if and only if every pair of implicants m_i and m_j has at least one literal in common.

(ii) *The property in* (i) *can be checked in $O(np^2)$ time.*

Proof. (i) $ff^* = (\bigvee_i m_i)(\bigvee_j m_j^*) = \bigvee_{i,j} m_i m_j^*$, where if $m_i = 23\bar{5}$ then $m_i^* = \bar{2}\bar{3}5$, and so on. Therefore, $ff^* = \perp$ (i.e., f is dual-minor by Lemma 3.3 (i)) if and only if $m_i m_j^* = \perp$ for all i and j, which holds if and only if every pair of implicants m_i and m_j has a common literal.

(ii) Immediate from (i). ∎

Now we cite the following well known NP-complete problems [16, 28].

Problem SAT (Satisfiability).

Input: A general function ϕ of n variables in CNF.

Question: Is $\phi \neq \perp$?

Problem NAE-SAT (Not-All-Equal SAT).

Input: A general function ϕ of n variables in CNF.

Question: Is there a vector $v \in \{0, 1\}^n$ such that $\phi(v) = \phi(\bar{v}) = 1$ (i.e., $\phi\phi^* \neq \perp$)?

The problem NDMAJOR for a general function f is NP-complete, since it is already NP-complete for positive functions (Theorem 3.2). However, we repeat this result below since its proof reveals an interesting equivalence with NAE-SAT.

THEOREM 3.4. *The problem of deciding if a given function f is not dual-major is* NP*-complete.*

Proof. Given an instance ϕ of NAE-SAT, consider the corresponding $f = \phi^d$ in DNF. f is obtained from ϕ by simply interchanging $\vee$ (or) and $\cdot$ (and). Obviously $\phi = f^d$ and $\phi^* = \bar{f}$. This ϕ is NAE-satisfiable if and only if $\phi\phi^* \neq \perp$. Then

$$\phi\phi^* \neq \perp \Leftrightarrow f^d\bar{f} \neq \perp \Leftrightarrow f^d \not\leqslant f \text{ (i.e., } f \text{ is not dual-major).}$$

Since our problem is obviously in class NP, this proves the theorem. ∎

Up to this point, the complexity for positive functions and general functions do not differ. However, the next result may indicate a difference between them.

THEOREM 3.5. *The problem of deciding if a dual-minor function f is not dual-major (i.e., f is not self-dual) is NP-complete.*

Proof. Let ϕ be an instance of problem SAT. Let $f = \phi^d$ (hence $f^* = \bar{\phi}$), where f and f^* are in DNF. Introduce $h = ff^*$, which is anti-dual as easily shown from definition. An expression of h in DNF can be computed in $O(np^2)$ time if f (and hence f^*) has p implicants. By Lemma 3.3 (ii), h can be represented as $h = g \vee g^*$ for a dual-minor function g. Then

$$\phi \neq \perp \Leftrightarrow h \neq \top \text{ (since } f \neq \top \text{ and } f^* \neq \top \text{ if and only if } \phi \neq \perp)$$
$$\Leftrightarrow g \vee g^* \neq \top \Leftrightarrow g^d \not\leqslant g;$$

i.e., ϕ is satisfiable if and only if a dual-minor function g is not dual-major. Since our problem is obviously in class NP, this completes the proof. ∎

4. INCREMENTALLY POLYNOMIAL ALGORITHMS

4.1. *Problems*

We return to positive functions. In conjunction with duality and self-duality, consider the following problems.

Problem DUALIZATION.

Input: A positive function f of n variables, i.e., min $T(f)$.

Output: Its dual $g = f^d$, i.e., min $T(f^d)$.

Problem SELF-DUALIZATION.

Input: A positive dual-minor function f of n variables, i.e., min $T(f)$.

Output: A positive self-dual function g (i.e., min $T(g)$) such that $f \leqslant g \leqslant f^d$.

Recall that the inputs to these problems are described by min $T(f)$ (i.e., prime implicants of f), and outputs are also given by min $T(g)$. Therefore, algorithms to solve these problems are enumerative in nature, i.e., have to enumerate all vectors in min $T(g)$. This means that, since the length of f^d can be exponential in the length of f, as noted in Section 3.1, such enumeration algorithms cannot be polynomial in the length of the input. In other words, what is interesting is the incremental polynomiality as explained in Section 1.

We show in this section that incrementally polynomial algorithms for DUALIZATION and SELF-DUALIZATION exist if and only if the corresponding problems MD and SD can be solved in polynomial time, respectively. The relation of these results with the original problem of IDENTIFICATION will also become clear.

4.2. *Algorithm for* DUALIZATION

We first consider Problem DUALIZATION. Assume that a set TD ($\subseteq \min T(f^d)$) has been enumerated so far, and define a positive function g by

$$\min T(g) = TD. \tag{14}$$

Then $g \leqslant f^d$ by $TD \subseteq \min T(f^d)$, and $g = f^d$ holds if and only if f and g are mutually dual. This mutual duality can be checked by solving problem MD for f and g. Furthermore, if $g \neq f^d$ (i.e., $g < f^d$), we can resort to algorithm UNKNOWN of Section 1 with

$$MT = \min T(f) \quad \text{and} \quad MF = \{\bar{v} \mid v \in TD\},$$

where $T(MT) \cap F(MF) = \varnothing$ by $g < f^d$ (recall relation (10)), to obtain an unknown vector $u \in \{0, 1\}^n \setminus (T(MT) \cup F(MF))$. This u satisfies $u \notin T(MT)$ (i.e., $u \in F(f)$) and $u \notin F(MF)$. From u, compute a maximal false vector $y \in \max F(f)$ by algorithm MAXIMAL and let $w := \bar{y}$. Then $y \in \max F(f)$ implies $w \in \min T(f^d)$, and $u \notin F(MF)$ implies $y \notin \max F(MF)$, i.e., $w \notin TD$. From these observations, we see that the following algorithm can be used to solve the Problem DUALIZATION.

ALGORITHM DUALIZE.

Input: A positive function f, i.e., min $T(f)$.

Output: min $T(f^d)$.

1. Start with $MT := \min T(f)$ and $TD := \varnothing$.

2. Solve problem MD with f and g, where g is defined by (14). If f and g are mutually dual, then output TD and halt. Otherwise go to 3.

3. Let $MF := \{\bar{v} \mid v \in TD\}$. Execute Algorithm UNKNOWN with MT and MF to obtain an unknown vector $u \in F(f)$. Then compute a maximal vector y satisfying $y \geqslant u$ and $y \in F(f)$ by Algorithm MAXIMAL, and let $w := \bar{y}$. Let $TD := TD \cup \{w\}$ and return to 2.

The time to solve problem MD in Step 2 is $T_{EQ}(n(|MT| + |MF|))$, as obvious from the reduction of MD into EQ in the proof (b) of Theorem 3.1. In Step 3, Algorithm UNKNOWN requires $O(nT_{EQ}(n(|MT| + |MF|)))$ time, as discussed in Section 2. Algorithm MAXIMAL then requires $O(n^2 |MT|)$ time, since instead of using membership queries, we directly compute the value of $f(y[y_i = 1])$ by the property: $f(y[y_i = 1]) = 1$ if there is $v \in MT$ such that $v \leqslant y[y_i = 1]$ and 0 otherwise. The time for checking the last condition is $O(n|MT|)$. Consequently, one iteration of Steps 2 and 3 is done in $O(nT_{EQ}(n(|MT| + |MF|)))$ time (assuming $T_{EQ}(n(|MT| + |MF|)) \geqslant O(n(|MT| + |MF|))$), and the total time of DUALIZE is

$$O(n\,|\min T(f^d)|\; T_{EQ}(n(|\min T(f)| + |\min T(f^d)|))).$$

Therefore, DUALIZE is incrementally polynomial if Problem MD (i.e., Problem EQ) can be solved in polynomial time.

THEOREM 4.1. *Problem* DUALIZATION *has an incrementally polynomial algorithm if and only if problem* MD *can be solved in polynomial time.*

Proof. The if-part was already proved above. To prove the converse, apply the incrementally polynomial Algorithm A for DUALIZATION to f until either (i) it halts or (ii) it outputs $|\min T(g)| + 1$ vectors, in order to solve Problem MD with input data f and g. In case of (i), we obtain min $T(f^d)$ and hence we can directly check if $g = f^d$ holds or not. In case of (ii), $g \neq f^d$ is immediately concluded. In any case, Problem MD is solved. The time required for this is $O(|\min T(g)|\, p(n(|\min T(f)| + |\min T(g)|)))$, where $p(\cdot)$ is the time for one iteration of Algorithm A. This is polynomial in the input length $n(|\min T(f)| + |\min T(g)|)$ of Problem MD. ∎

Since problem DUALIZATION is a fundamental topic in Boolean theory [4], its relation with Problem MD has been recognized in various papers including [10, 12, 26]. The above argument may be interesting in the sense that

it reveals a similarity between Algorithms DUALIZE and IDENTIFY. Although DUALIZE starts with $MT := \min T(f)$ and $MF(=\{\bar{v} \mid v \in TD\}) := \emptyset$, IDENTIFY starts with $MT := MF := \emptyset$. DUALIZE then computes the final $MF = \max F(f)$ from the knowledge of $MT = \min T(f)$, but IDENTIFY computes both $MT = \min T(f)$ and $MF = \max F(f)$ by issuing membership queries to the oracle of f. The iteration of both algorithms to find a new vector is based on essentially the same idea, as evident from the above discussion.

4.3. *Algorithm for* SELF-DUALIZATION

For a dual-minor function f, the whole set $\{0, 1\}^n$ is partitioned into three disjoint sets, $T(f)$, $T(f^d\bar{f})(=T(f^d)\backslash T(f))$ and $F(f^d)$. Furthermore, $T(f)$ and $F(f^d)$ are related by (10). Therefore, (a) $v \in T(f)$ if and only if $\bar{v} \in F(f^d)$, and (b) $v \in T(f^d\bar{f})$ if and only if $\bar{v} \in T(f^d\bar{f})$. Also condition (10) implies

$$\max F(f^d) = \{\bar{v} \mid v \in \min T(f)\}. \tag{15}$$

Without loss of generality, we assume $f \neq \perp$ in this section, since otherwise any self-dual function can be output.

For a dual-minor function f, a vector v is called a *counterexample* to f if $f(v) = f(\bar{v}) = 0$, since this indicates that f is not self-dual (see the characterizations (i) and (iii) in Section 3.1). The self-duality can be checked by solving Problem SD. If f is not self-dual, i.e., $f < f^d$, then a vector u satisfying

$$u \in \{0, 1\}^n \backslash (T(f) \cup F(f^d)) \tag{16}$$

can be found by Algorithm UNKNOWN executed with

$$MT = \min T(f) \quad \text{and} \quad MF = \{\bar{v} \mid v \in \min T(f)\},$$

where $T(MT) \cap F(MF) = T(f) \cap F(f^d) = \emptyset$ by assumption $f < f^d$. This u is a counterexample, i.e., $f(u) = f(\bar{u}) = 0$, because $u \in T(f^d\bar{f})$ by (16) and hence $\bar{u} \in T(f^d\bar{f})$ by property (b) above.

Given a counterexample u, we then obtain a minimal vector $y \leq u$ such that $y \in T(f^d)$ by applying MINIMAL, in which condition $y'(=y[y_i = 0]) \in T(f^d)$ in Step 2 of MINIMAL is checked by condition $y' \notin F(f^d)$ (i.e., there is no vector $v \in \max F(f^d)$ such that $v \geq y'$) in $O(n\,|MT|)$ time, since $\max F(f^d)$ is provided by (15) instead of the explicit data of $\min T(f^d)$. We then define a new function g satisfying $f < g \leq f^d$ by

$$\min T(g) = \text{MinSet}(\min T(f) \cup \{y\}).$$

We claim that this g is positive and dual-minor. The positivity is an immediate consequence from definition. To show that g is dual-minor, assume otherwise, i.e., there is a pair of vectors w and $\bar{w}$ such that $g(w) = g(\bar{w}) = 1$. These vectors belong to $T(f^d\bar{f})$ by the above properties (a) and (b), and hence satisfy $w \geq y$ and $\bar{w} \geq y$ by the above definition of g. But this implies $y = (00\cdots0)$, i.e., $g = \top$, which is a contradiction to assumption $f \neq \perp$. Therefore, g is positive and dual-minor.

The following algorithm repeats the above procedure until a self-dual function is obtained.

ALGORITHM SELF-DUALIZE.

Input: A positive dual-minor function f, i.e., $\min T(f)$.

Output: A positive self-dual function g, i.e., $\min T(g)$, such that $f \leq g \leq f^d$.

1. Start with $MT := \min T(f)$ and $g := f$.

2. Solve Problem SD with g. If g is self-dual, then output MT and halt. Otherwise go to 3.

3. Let $MF := \{\bar{v} \mid v \in MT\}$. Execute Algorithm UNKNOWN with MT and MF to obtain an unknown vector $u \in \{0, 1\}^n \backslash (T(g) \cup F(g^d))$. Then compute a minimal vector y such that $y \leq u$ and $y \in T(g^d) \backslash T(g)$ by algorithm MINIMAL, and let $MT := \text{MinSet}(MT \cup \{y\})$. Define the function g by $\min T(g) = MT$. Return to 2.

First note that, by their minimality with respect to $T(f^d)$, all the y's generated in Step 3 will remain in the final output, i.e., $\min T(g)$. Hence we may regard that such y is successively output in each iteration, and the rest, i.e., $\min T(f) \cap \min T(g)$, are output in the last iteration. Similarly to Steps 2 and 3 of DUALIZE, SELF-DUALIZE requires

$$O(nT_{EQ}(n\,|MT|))(\leq O(nT_{EQ}(n(|\min T(f)| + |Y|))))$$

time in each iteration, where Y is the set of y's generated by then. Therefore, this is an incrementally polynomial algorithm if problem SD (i.e., problem EQ) can be solved in polynomial time. The total time is

$$O(n\,|\min T(g)|\,T_{EQ}(n(|\min T(f)| + |\min T(g)|))),$$

where g in this formula is the last g computed by SELF-DUALIZE.

THEOREM 4.2. *Problem* SELF-DUALIZATION *has an incrementally polynomial algorithm if and only if problem* SD *can be solved in polynomial time.*

Proof. The if-part follows from the above argument. To prove the only-if-part, first check if f is dual-minor (which can be done in polynomial time by Lemma 3.1). If f is not dual-minor, it is not self-dual. If f is dual-minor, apply the incrementally polynomial Algorithm A for SELF-DUALIZATION until either (i) it halts or (ii) it iterates $|\min T(f)| + 1$ times. In case (i), f is self-dual if and only if

the obtained function g satisfies $g = f$. In case (ii), f is not self-dual, since each vector y generated by A belongs to min $T(g)$. In any case, problem SD is solved. The time required for this is $O(|\min T(f)|\, p(n\,|\min T(f)|))$, where $p(\cdot)$ is the time for one iteration of Algorithm A. This is polynomial in the input length $n(|\min T(f)|)$ of problem SD. ∎

For a given dual-minor function f, there are in general many self-dual functions g satisfying $f \leqslant g \leqslant f^d$. The above Algorithm SELF-DUALIZE obtains only one of them. If other unknown vectors are used in Step 3 (e.g., $\bar{u}$ instead of u), the resulting function g may become different.

Combining Theorems 4.1 and 4.2 with Theorem 3.1, we obtain the next result.

COROLLARY 4.1. *Three problems,* IDENTIFICATION, DUALIZATION, *and* SELF-DUALIZATION, *are equivalent in the sense that, if one of them has an incrementally polynomial algorithm, then so do the rest of them.*

4.4. *Special Classes of Positive Functions*

It is often known that, for special classes of positive functions, some of the above problems IDENTIFICATION, DUALIZATION, and SELF-DUALIZATION are polynomially solvable. However, it does not automatically mean that the rest of the problems are also polynomially solvable for such special classes, since technical details used to show the equivalence in Corollary 4.1 must be checked. For simplicity, we concentrate here on problem IDENTIFICATION, and give some sufficient conditions for a subclass C of positive functions to have an incrementally polynomial algorithm. We say that a class C of positive functions is *closed under fixing of variables* if $f \in C$ implies $f_a \in C$ for any $a = (a_1, a_2, ..., a_k)$.

(A) For any $MT \subseteq \min T(f)$ and $MF \subseteq \max F(f)$ of a function $f \in C$, Problem EQ can be solved in polynomial time, and, if $T(MT) \cup F(MF) \neq \{0, 1\}^n$, an unknown vector u can be computed in polynomial time. (In this case, it is immediate to see that Steps 2 and 3 of IDENDIFY can be executed in polynomial time.)

(B) C is closed under fixing of variables, and it can be determined in polynomial time whether $f \in C$ holds or not. Also Problem DUALIZATION for $f \in C$ has an incrementally polynomial algorithm. (In this case, in Step 2 of IDENTIFY, define f' by $\min T(f') = MT$. If $f' \notin C$, then EQ has answer "no," and if $f' \in C$, compute $(f')^d$ to solve EQ in the manner of Theorem 3.1(b) and Theorem 4.1. This can be done in polynomial time. Step 3 of IDENTIFY can also be executed similarly in polynomial time.)

(C) C is closed under fixing of variables, and it can be determined in polynomial time whether $f \in C$ holds or not. Also Problem MD for f and g, one of which belongs to class C, can be solved in polynomial time. (In this case, Algorithm DUALIZE is incrementally polynomial. Then apply (B).)

For example, there are incrementally polynomial DUALIZATION algorithms for such classes as positive threshold functions and regular functions [26, 10, 5], positive k-DNF functions [12], and matroid functions [23]. These classes are closed under fixing of variables, and it can be determined in polynomial time whether $f \in C$ holds or not. Therefore, by (B), IDENTIFICATION for these classes has incrementally polynomial algorithms.

5. SATURATED POSITIVE FUNCTIONS

In the course of investigating the complexity of Problems MD and SD, some other problems have been found to be polynomially equivalent. We list one problem, extensively studied in [12] in the hypergraph setting. A family $\mathscr{C}$ of subsets of $V = \{1, 2, ..., n\}$ is sometimes referred to as a *hypergraph* $H = (V, \mathscr{C})$, where each $S \in \mathscr{C}$ is called a hyperedge. A hypergraph H is *simple* if $\mathscr{C}$ is incomparable.

Problem SIMPLE-H-SAT (Simple Hypergraph Saturation).

Input: A simple hypergraph $H = (V, \mathscr{C})$.

Question: Is there no subset S of V such that $H' = (V, \mathscr{C} \cup \{S\})$ is also a simple hypergraph?

In [12], it is shown that this problem is polynomially equivalent to Problem SD. It also contains interesting applications found in such fields as artificial intelligence and database theory. In this section, we define the same problem in Boolean terminology and again show its polynomial equivalence with SD. As the methods of Boolean algebra become available in this way, the proof is much simplified and easy to understand. Some new characterizations such as Lemma 5.1(iii) below are also derived. Furthermore, some concepts introduced in its proof turn out to be very useful to understand properties of self-dual functions, as will be discussed in a companion paper [8].

For a positive function f, call a vector $w \in \{0, 1\}^n$ *incomparable* with f if $w \not\leqslant v$ and $w \not\geqslant v$ hold for all $v \in \min T(f)$. A positive function f is *saturated* if there is no vector w incomparable with f.

Problem ST.

Input: A positive function f of n variables, i.e., $\min T(f)$.

Question: Is f saturated?

Before proving the polynomial equivalence of ST and SD, we introduce two positive functions f^s and f_c associated with f. First f_c is defined by

$$\min T(f_c) = \{\bar{v} \mid v \in \min T(f)\}. \tag{17}$$

The DNF expression of f_c by prime implicants is

$$f_c = \bigvee \{m_v \mid v \in \min T(f)\}, \tag{18}$$

where m_w for $w \in \{0, 1\}^n$ denotes the conjunction of literals x_i satisfying $w_i = 1$; e.g., if $w = (01011)$ then $m_w = 245$ (abbreviation of $x_2x_4x_5$). For example,

$$\begin{aligned} f &= 12 \vee 234, \quad \text{i.e.,} \quad \min T(f) = \{(1100), (0111)\} \\ f_c &= 34 \vee 1, \quad \text{i.e.,} \quad \min T(f_c) = \{(0011), (1000)\}. \end{aligned} \tag{19}$$

Next, let $L(w)$ denote the set of vectors obtained from w by changing one element $w_i = 0$ to $w_i = 1$. For example, for $w = (01001)$, $L(w) = \{(11001), (01101), (01011)\}$. Function f^s, which is called the *subdual* of f, is then defined by

$$\min T(f^s) = \text{MinSet}\left(\bigcup \{L(\bar{v}) \mid v \in \min T(f)\}\right). \tag{20}$$

Its DNF expression is given by

$$f^s = \bigvee \{m_v m_v^d \mid v \in \min T(f)\}, \tag{21}$$

where m_v^d denotes the dual of m_v; e.g., if $m_v = 235$ then $m_v^d = (2 \vee 3 \vee 5)$. For the example of (19), we have

$$\begin{aligned} f^s &= 34(1 \vee 2) \vee 1(2 \vee 3 \vee 4) = 234 \vee 12 \vee 13 \vee 14 \\ \min T(f^s) &= \text{MinSet}\{(1011), (0111), (1100), (1010), (1001)\} \\ &= \{(0111), (1100), (1010), (1001)\}. \end{aligned} \tag{22}$$

Now recall property (15) that holds between $\max F(f^d)$ and $\min T(f)$. Then the definition of f_c implies $\min T(f_c) = \max F(f^d)$. Since any $w \in \min T(f^s)$ is larger than some $y \in \max F(f^d)$, by definitions (17) and (20), it satisfies $w \not\leqslant z$ for all $z \in \max F(f^d)$, implying $w \in T(f^d)$. This proves

$$f^s \leqslant f^d, \qquad \text{i.e.,} \quad (f^s)^d \geqslant f. \tag{23}$$

It is also clear from definitions (18) and (21) that

$$f^s \leqslant f_c. \tag{24}$$

Therefore $f^s \leqslant f^d f_c$. The converse is also true since

$$\begin{aligned} f^d f_c &= \bigvee_{v \in \min T(f)} m_v \left(\bigvee_{w \in \min T(f)} m_w \right)^d \\ &= \bigvee_v m_v \left(\bigwedge_w m_w^d \right) \\ &\leqslant \bigvee_v m_v m_v^d = f^s, \end{aligned}$$

which proves

$$f^s = f^d f_c. \tag{25}$$

LEMMA 5.1. *For a positive function f, the following conditions are equivalent*:

(i) *f is saturated.*

(ii) $f^d \leqslant f_c$.

(iii) $f^s = f^d$.

Proof. A vector w is incomparable with f if and only if (a) $w \notin T(f)$, and (b) $w \wedge \bar{v} \neq (00\cdots0)$ (i.e., $w \not\leqslant v$) for all $v \in \min T(f)$. However, condition (b) is equal to saying that $w \in T((f_c)^d)$ (see definition (17) of f_c and the transversality of a dual function, mentioned in Section 3.1). Therefore, such an incomparable w does not exists if and only if

$$T(f) \supseteq T((f_c)^d) \Leftrightarrow f \geqslant (f_c)^d \Leftrightarrow f^d \leqslant f_c.$$

This proves the equivalence of (i) and (ii). The equivalence of (ii) and (iii) is then obvious from property (25). ∎

For example, function f of (19) is not saturated, since $f^d = (1 \vee 2)(2 \vee 3 \vee 4) = 2 \vee 13 \vee 14$ is not equal to f^s of (22). In fact, a vector (1011) is incomparable with f.

COROLLARY 5.1. *If f is a saturated positive function of n variables, its dual f^d satisfies* $|\min T(f^d)| \leqslant n\,|\min T(f)|$, *and* $\min T(f^d)$ *can be computed* (*i.e.*, DUALIZATION *can be solved*) *in polynomial time.*

Proof. Immediate from property $f^s = f^d$ in Lemma 5.1(iii), and expression (21) of f^s. ∎

However, this does not say that Problem ST can be solved in polynomial time. It is in fact polynomially equivalent to Problem SD, as noted in the next theorem. Other problems such as EQ and IDENTIFICATION for saturated positive functions can also be shown to have the same complexity as those for general positive functions.

THEOREM 5.1. *Problems* ST *and* SD *are polynomially equivalent.*

Proof. (a) Reduction from ST to SD. Let f be a positive function, and x, y be variables not contained in f. From this f, construct the following h:

$$h = fx \vee f^s y \vee xy;$$

i.e., by (13),

$$h^d = (f^s)^d x \vee f^d y \vee xy.$$

Then (23) shows that h is dual-minor, i.e., $h \leqslant h^d$, and furthermore $h = h^d$ holds if and only if $f^s = f^d$. Thus, h is self-dual if and only if f is saturated.

(b) Reduction from SD to ST. Assume that f is positive and dual-minor (dual-minority can be checked in polynomial time by Lemma 3.1). From f, define

$$g = f \vee f_c xy, \tag{26}$$

where x and y are variables not contained in f. Then

$$\min T(g) = \{(v00), (\bar{v}11) \mid v \in \min T(f)\}, \tag{27}$$

where the last two elements of vectors respectively correspond to x and y, because $f \leqslant f^d$ implies $v \wedge w \neq (00\cdots0)$ for any $v, w \in \min T(f)$ (by Lemma 3.1), and hence $v \not\leqslant \bar{w}$ for any $v, w \in \min T(f)$. Property (27) says that $w \in \min T(g) \Leftrightarrow \bar{w} \in \min T(g)$, and hence $g = g_c$. Therefore,

$$\begin{aligned} g \text{ is saturated} &\Leftrightarrow g^d \leqslant g_c \qquad \text{(by Lemma 5.1(ii))} \\ &\Leftrightarrow g^d \leqslant g \\ &\Leftrightarrow f^d((f_c)^d \vee x \vee y) \leqslant f \vee f_c xy. \end{aligned}$$

The last inequality implies $f^d \leqslant f$ (take $x = 1$ and $y = 0$); i.e., $f^d = f$ (self-dual) by assumption $f \leqslant f^d$. On the other hand, since $f \leqslant g$ (i.e., $g^d \leqslant f^d$) follows from (26), $f^d \leqslant f$ implies $g^d \leqslant f^d \leqslant f \leqslant g = g_c$; i.e., g is saturated by Lemma 5.1(ii). Consequently, g is saturated if and only if f is self-dual. ∎

6. CONCLUSION

We have shown in this paper that all the following questions are equivalent in the sense that either all hold true or none:

(1) EQ $\in P$,

(2) MD $\in P$,

(3) SD $\in P$,

(4) ST $\in P$,

(5) IDENTIFICATION $\in$ *IncrP*,

(6) IDENTIFICATION $\in$ *TotalP*,

(7) DUALIZATION $\in$ *IncrP*,

(8) SELF-DUALIZATION $\in$ *IncrP*.

Here P stands for the class of problems solvable in polynomial time, *IncrP* the class solvable by incrementally polynomial time algorithms, and *TotalP* the class solvable by polynomial total time algorithms. Although some important steps have been made [13], their complexity is still left open.

As noted in Section 4.4, there are several special classes of positive functions, for which polynomial (or incrementally polynomial) algorithms are known. One of our companion papers [23] investigates this direction by using a new tool called the maximum latency, which is a complexity measure for finding an unknown vector, and provides some polynomially solvable classes. Another possible avenue would be to introduce approximate concepts of dual, self-dual, saturation and so forth, so that they become computationally tractable. The concept of subdual introduced in Section 5 is interesting in this respect, since it is a subfunction of the dual and is polynomially computable for all positive functions. The concept of *almost self-duality*, which is a close approximation of the self-duality defined in terms of subduality (hence is polynomially testable), and related topics are extensively studied in another companion paper [8].

ACKNOWLEDGMENTS

The authors have benefited from the discussion with their colleagues E. Boros, Y. Crama, T. Eiter, P. L. Hammer, T. Kameda, K. Makino, H. Nagamochi, and others, whose cooperation is greatly appreciated. This paper was partly completed while the first author visited Kyoto University in April 1993, the support for which is greatly appreciated. This research was partially supported by the Scientific Grant-in-Aid from Ministry of Education, Science and Culture of Japan.

Received June 21, 1994; final manuscript received March 1, 1995

REFERENCES

1. Angluin, D. (1988), Queries and concept learning, *Mach. Learning* **2**, 319–342.
2. Angluin, D., Hellerstein, L., and Karpinski, M. (1993), Learning read-once formulas with queries, *J. Assoc. Comput. Mach.* **40**, 185–210.
3. Anthony, M., and Biggs, N. (1992), "Computational Learning Theory," Cambridge Univ. Press, Cambridge, UK.
4. Benzaken, C. (1966), Algorithme de dualisation d'une fonction booléenne, *Rev. Fr. Traitment Inform. Chiffres* **9**, 119–128.
5. Bertolazzi, P., and Sassano, A. (1987), An $O(mn)$ time algorithm for regular set-covering problems, *Theoret. Comput. Sci.* **54**, 237–247.
6. Bioch, J. C. (1993), "Decompositions of Coteries and Boolean Functions," Tech. Rep. Dept. of Computer Science No. 2, Erasmus University Rotterdam, The Netherlands.
7. Bioch, J. C., and Ibaraki, T. (1995), Decompositions of positive self-dual Boolean functions, *Discrete Math.* **140**, 23–46.
8. Bioch, J. C., and Ibaraki, T. (1994), "Generating and Approximating Positive Nondominated Coteries," Technical Report RRR 41-94, RUTCOR, Rutgers University; *IEEE Trans. Parallel Distrib. Systems*, to appear.
9. Boros, E., Hammer, P. L., Ibaraki, T., and Kawakami, K. (1991), Identifying 2-monotonic positive Boolean functions in polynomial time, *in* "ISA'91, Algorithms" (W. L. Hsu and R. C. T. Lee, Eds.), Lecture Notes in Computer Science, Vol. 557, pp. 104–115, Springer-Verlag, Berlin/New York; *SIAM J. Comput.*, to appear.
10. Crama, Y. (1987), Dualization of regular Boolean functions, *Discrete Appl. Math.* **16**, 79–85.
11. Crama, Y., Hammer, P. L., and Ibaraki, T. (1988), Cause-effect relationships and partially defined Boolean functions, *Ann. Oper. Res.* **16**, 299–326.
12. Eiter, T., and Gottlob, G. (1991), "Identifying the Minimal Transversals of a Hypergraph and Related Problems," Technical Report CD-TR 91/16, Christial Doppler Labor für Expertensysteme, Technische Universität Wien; *SIAM J. Comput.*, to appear.

13. Fredman, M., and Khachiyan, L. (1994), "On the Complexity of Dualization of Monotone Disjunctive Normal Forms," Technical Report LCSR-TR-225, Department of Computer Science, Rutgers University.
14. Gainanov, D. N. (1984), On one criterion of the optimality of an algorithm for evaluating monotonic Boolean functions, *U.S.S.R. Comput. Math. and Math. Phys.* **24**, 176–181.
15. Garcia-Molina, H., and Barbara, D. (1985), How to assign votes in a distributed system, *J. Assoc. Comput. Mach.* **32**, 841–860.
16. Garey, M. R., and Johnson, D. S. (1979), "Computers and Intractability," Freeman, New York.
17. Halmos, P. R. (1963), "Lectures on Boolean Algebras," Van Nostrand, Princeton, NJ.
18. Ibaraki, T., and Kameda, T. (1993), A theory of coteries: Mutual exclusion in distributed systems, *IEEE Trans. Parallel Distrib. Systems* **4**, 779–794.
19. Johnson, D. S., Yannakakis, M., and Papadimitriou, C. H. (1988), On generating all maximal independent sets, *Inform. Process. Lett.* **27**, 119–123.
20. Kavvadias, D., Papadimitriou, C. H., and Sideri, M. (1993), On Horn envelopes and hypergraph transversals, *in* "ISAAC'93, Algorithms and Computation" (K. W. Ng, P. Raghavan, N. V. Blasubramanian and F. Y. L. Chin, Eds.), Lecture Notes in Computer Science, Vol. 762, pp. 399–405, Springer-Verlag, Berlin/New York.
21. Lamport, L. (1978), Time, clocks, and the ordering of events in a distributed system, *Comm. ACM* **21**, 558–565.
22. Lawler, E., Lenstra, J. K., and Rinnooy Kan, A. H. G. (1980), Generating all maximal independent sets: NP-hardness and polynomial-time algorithms, *SIAM J. Comput.* **9**, 558–565.
23. Makino, K. and Ibaraki, T. (1994), The maximum latency and identification of positive Boolean functions, *in* "ISAAC'94, Algorithms and Computation" (D. Z. Du and X. S. Zhang, Eds.), Lecture Notes in Computer Science, Vol. 834, pp. 324–332, Springer-Verlag, Berlin/New York.
24. Mannila, H., and Räihä, K.-J. (1986), Design by example: An application of Armstrong relations, *J. Comput. System Sci.* **22**, 126–141.
25. Muroga, S. (1971), "Threshold Logic and Its Applications," Wiley-Interscience, New York.
26. Peled, U. N., and Simeone, B. (1985), Polynomial-time algorithm for regular set-covering and threshold synthesis, *Discrete Appl. Math.* **12**, 57–69.
27. Reiter, R. (1987), A theory of diagnosis from first principle, *Artif. Intell.* **32**, 57–95.
28. Schaefer, T. J. (1978), The complexity of satisfiability problems, *in* "Proc. 10th Ann. ACM Symp. on Theory of Computing, New York," pp. 216–226.
29. Valiant, L. G. (1984), A theory of the learnable, *Comm. ACM* **27**, 1134–1142.
30. Wegener, I. (1987), "The Complexity of Boolean Functions," Wiley-Teubner, New York.