The Query Complexity of Finding Local Minima in the Lattice[1]

Amos Beimel[2]
Dept. of Computer Science, Ben-Gurion University, Beer-Sheva 84105, Israel.
E-mail: beimel@cs.bgu.ac.il.
Homepage: www.cs.bgu.ac.il/∼beimel.

Felix Geller
Computer Science Department, Technion Haifa 32000, Israel.
E-mail: felix@cs.technion.ac.il.

and

Eyal Kushilevitz[3]
Computer Science Department, Technion Haifa 32000, Israel.
E-mail: eyalk@cs.technion.ac.il.
Homepage: www.cs.technion.ac.il/∼eyalk.

Received ; accepted Oct, 2000

In this paper we study the query complexity of finding local minimum points of a boolean function. This task occurs frequently in exact learning algorithms for many natural classes, such as monotone DNF, $O(\log n)$-term DNF, unate DNF, and decision trees. On the negative side, we prove that any (possibly randomized) algorithm that produces a local minimum of a function f chosen from a sufficiently "rich" concept class, using a membership oracle for f, must ask $\Omega(n^2)$ membership queries in the worst case. In particular, this lower bound applies to the class of decision trees. A simple algorithm is known that achieves this lower bound. On the positive side, we show that for the class $O(\log n)$-term DNF finding local minimum points requires only $\Theta(n \log n)$ membership queries (and more generally $\Theta(tn)$ membership queries for t-term DNF with $t \leq n$). This efficient procedure improves the time and query complexity of known learning algorithms for the class $O(\log n)$-term DNF.

---

[1] An early version of this paper appeared in the proceedings of the 11th Conference on Computational Learning Theory (COLT), pp. 294-302, July 1998.
[2] This work was done while the author was with the division of engineering & applied sciences at Harvard University, supported by grants ONR-N00014-96-1-0550 and ARO-DAAL-03-92-G0115.
[3] Supported by Technion V.P.R. Fund 120-872, by Japan Technion Society Research Fund, by the fund for the promotion of research at the Technion, and by the German-Israeli Foundation for scientific research and development (GIF).

1. INTRODUCTION

Angluin's model of learning using membership queries and equivalence queries (i.e., the *exact learning model*) [3] attracted a lot of attention. In particular, various concept classes were shown to be learnable in this model (e.g., [1, 2, 5, 12, 13, 26, 28, 14, 10, 9] and many others). In some of the above, and in several related papers (e.g., [3, 18, 6, 13, 8, 11, 14, 7]), the following common approach is used: the input space, $\{0,1\}^n$, is viewed as a lattice with the natural partial order, i.e., for $u, v \in \{0,1\}^n$ if $u[i] \leq v[i]$ for all i then $u \leq v$ (where $u[i]$ is the i-th bit of u). Then, the learning algorithm collects information about the target function f by repeatedly looking for *minimal points* of f in the lattice; that is, points u such that $f(u) = 1$ but for each v, a direct "child" of u in the lattice, $f(v) = 0$. For finding such points, the learning algorithm uses a procedure FIND_TERM that gets as an input some point w such that $f(w) = 1$ and searches for a minimal point u such that $u \leq w$. Angluin [3], considering the case of *monotone* functions, implemented the procedure FIND_TERM using n membership queries as follows:

```
For 1 ≤ i ≤ n:
    If w[i] = 1 and when flipping w[i] to 0 the value of f remains 1
                  then flip w[i] to 0 and proceed with the modified w.
End-For
The last value of w is a minimal point.
```

Bshouty [13, 14] considered *non-monotone* classes of functions (such as decision trees), and implemented the procedure FIND_TERM using $O(n^2)$ membership queries as follows:

```
Repeat (at most n times):
    Find an index 1 ≤ i ≤ n such that
        w[i] = 1 and when flipping w[i] to 0 the value of f remains 1.
    Flip w[i] to 0 and proceed with the modified w.
Until no such i is found
The last value of w is a minimal point.
```

This procedure finds a local minimum point for every boolean function (provided that it starts with a point w such that $f(w) = 1$).

Being a basic building block in many algorithms, which is used many times in each execution, it is important to study the complexity (i.e., number of membership queries) of finding minimal points.

Our main result is a negative result: we prove that any implementation of the procedure FIND_TERM with respect to "sufficiently rich" classes of (possibly non-monotone) boolean functions requires $\Omega(n^2)$ membership queries. This lower bound holds even if the implementation is randomized (in this case the *expected* number of queries is $\Omega(n^2)$). Furthermore, the lower bound remains true even if we restrict ourselves to the class of "small" (polynomial-size) decision trees, or to the class of "small" DNF formulae. Therefore, Bshouty's implementation for the non-monotone case is optimal; this should be contrasted with Angluin's $O(n)$ upper bound for the monotone case.

On the positive side, for the class t-term DNF we prove that $O(tn)$ membership queries are sufficient for implementing the procedure FIND_TERM. (A boolean function is in the class t-term DNF if it can be represented as a disjunction of at most t terms.) We show that after repeating Angluin's implementation $O(t)$ times, the resulting point is a minimal point. (Our procedure is described in detail in Fig. 2.) This efficient implementation of FIND_TERM improves the time and query complexity of learning the class $O(\log n)$-term DNF by a factor of $O(n/\log n)$ in Bshouty's algorithm based on the monotone theory [13] and in Bshouty's divide and conquer algorithm [14]. On the other hand, we prove a matching lower bound of $\Omega(tn)$ for finding minimal points for target functions from the class t-term DNF (for every $t \leq n$).

Related Work. The query complexity of learning algorithms was the subject of a considerable amount of work; we provide some examples below. Angluin [3] proved for several classes that polynomial number of membership queries do not suffice for learning if equivalence queries are not allowed. Angluin [4] showed the significance of membership queries for efficient learning of classes such as NFA, DFA, CFG, DNF, and CNF formulae. Maass and Turán [22, 23, 24] established general lower bounds, relating the query complexity of a concept class to combinatorial parameters of the class. Hegedüs [19] and Hellerstein et al. [20] further investigated this direction. They gave a combinatorial characterization of the number of queries required to learn a class by a (possibly non-efficient) algorithm that uses proper equivalence queries and membership queries. Hegedüs [19] also gave combinatorial characterization for algorithms that only use membership queries. Bshouty et al. [15, 16] investigated the number of equivalence queries required for learning, assuming that only polynomially many membership queries are available (in addition to the equivalence queries). Clausen et al. [17] and Roth and Benedek [27] investigated the number of membership queries required to interpolate polynomials over finite fields (without equivalence queries).

Organization. In Section 2 we present the definitions and notation which are used throughout the paper. In Section 3 we prove our lower bound for deterministic algorithms, and in Section 4 we prove the lower bound for randomized algorithms. In Section 5 we prove that $\Theta(tn)$ queries are necessary and sufficient for finding local minimum points for the class t-term DNF ($t \leq n$). Finally, in Appendix A we prove an asymptotically tight lower bound for randomized algorithms (improving the lower bound of Section 4 by a constant).

2. PRELIMINARIES

In this section we present some definitions and notation regarding the boolean lattice, and formally define the task of finding local minimum points.

We view the space $\{0,1\}^n$ as a lattice with the following partial order relation: for $u, v \in \{0,1\}^n$ define $u \leq v$ if for each i ($1 \leq i \leq n$) $u[i] \leq v[i]$, where $u[i]$ denotes the i-th bit of the vector u. We call u a *son* of v if $u \leq v$ and $u[i] < v[i]$ for *exactly* one index i. A point v is a *local minimum* of the boolean function f if $f(v) = 1$ but for each u, a son of v, $f(u) = 0$. For a boolean vector v, let weight(v) denote the number of non-zero coordinates of v. The weight function induces a partition of the boolean lattice into $n+1$ levels: the i-th level comprises all the points of weight exactly i.

To facilitate the presentation of our results, we use graph-theoretic terminology. We consider a directed graph, denoted $G_B(V, E)$, corresponding to the boolean lattice where $V = \{0,1\}^n$ (vertices represent boolean vectors), and $E = \{(v, u) : u \text{ is son of } v\}$ (edges exist between parents and sons; each edge is directed from a parent to its son). Given a path $p = \{v_0, v_1, \ldots, v_k\}$ in G_B, we define a boolean function f_p which is true only on the path vertices: $f_p(u) = 1 \iff u = v_i$ for some i ($0 \leq i \leq k$). The class $\mathcal{P}_n$ consists of all boolean functions which correspond to paths starting at the top vertex of the lattice (i.e., 1^n) and ending at a vertex on the second level:

$$\mathcal{P}_n = \left\{ f_p \;\middle|\; p = \{v_0, v_1, ..., v_{n-2}\} \text{ is a path in } G_B, \text{ and } v_0 = 1^n \right\}.$$

Note that, for any $p \in \mathcal{P}_n$, the function f_p has a unique minimal point (i.e, v_{n-2}) and that this point is located on the second level of the lattice (i.e., weight$(v_{n-2}) = 2$). On the other hand, each point on the second level of the lattice is a minimal point for many functions (more precisely, $(n-2)!$ functions).

We say that $\mathcal{A}$ produces local minimum points for the class $\mathcal{F}$ if for each $f \in \mathcal{F}$ the algorithm $\mathcal{A}$ on input u_0, such that $f(u_0) = 1$, and given an access to a membership oracle for f, halts in finite time and outputs u, a local minimum of f such that $u \leq u_0$. When u_0 is omitted, we assume that the starting point is 1^n. We denote the number of membership queries performed by (a deterministic) algorithm $\mathcal{A}$ on input f and u_0 by MC$(\mathcal{A}, f, u_0)$; the worst-case membership query complexity of $\mathcal{A}$ on $\mathcal{F}$ is

$$\text{MC}(\mathcal{A}, \mathcal{F}) = \max_{f \in \mathcal{F}, u_0 \in \{0,1\}^n \text{ s.t. } f(u_0)=1} \text{MC}(\mathcal{A}, f, u_0).$$

(These definitions are extended in Section 4 to deal with randomized algorithms.)

3. THE LOWER BOUND FOR DETERMINISTIC ALGORITHMS

In this section we prove a tight lower bound on the number of membership queries required for finding local minimum points using deterministic algorithms. (Essentially the same lower bound, for the more involved randomized case, is proved in Section 4.)

THEOREM 3.1. *For any deterministic algorithm $\mathcal{A}$ and any concept class $\mathcal{F}$, such that $\mathcal{P}_n \subseteq \mathcal{F}$,*

$$\text{MC}(\mathcal{A}, \mathcal{F}) \geq \binom{n}{2} - 1.$$

Many "natural" concept classes include all the functions in $\mathcal{P}_n$. For example, any $f \in \mathcal{P}_n$ has a compact DNF representation: $f \equiv \bigvee_{i=0}^{n-2} T_i$, where for each i,

$$T_i = \overline{x_{\sigma(1)}} \wedge \ldots \wedge \overline{x_{\sigma(i)}} \wedge x_{\sigma(i+1)} \wedge \ldots \wedge x_{\sigma(n)}$$

for some σ, a permutation of $\{1, 2, \ldots, n\}$ (σ is common to all the terms). Given the DNF formula for $f \in \mathcal{P}_n$, it is also straightforward to obtain a decision tree of size $O(n^2)$ for f: the tree tests the variables according to the order specified by the permutation σ: $x_{\sigma(1)}, x_{\sigma(2)}, \ldots, x_{\sigma(n)}$. Thus, $\mathcal{P}_n$ is properly included in the class of functions which have a decision tree representation of polynomial size.

An important point is that the lower bound applies to any strategy for finding minimal points even if unlimited computational power is used by the strategy. Another remark is that this lower bound is tight up to an *additive* factor of $O(\log n)$ since a simple modification of Bshouty's implementation [13] uses $\binom{n}{2} + \lceil \log n \rceil$ membership queries to find local minimum points for any boolean function (the naive implementation of [13] uses $\binom{n+1}{2} - 1$

membership queries).[4] In the rest of this section we restrict our attention to the sub-class $\mathcal{P}_n$.

What algorithms that produce the minimal point of a function in $\mathcal{P}_n$ could we conceive? As noted in the Introduction, one approach is to start from u_0 and go down the lattice towards the minimal point (we denote this algorithm by $\mathcal{A}_{\text{flip}}$). Another plausible option is to go over all the points of the second level and to find the point on which the function assumes the value "1" (call this implementation $\mathcal{A}_{\text{obl}}$). Clearly, both procedures require $\Omega(n^2)$ membership queries. Still, it could be the case that some sophisticated algorithm performs asymptotically better than the two "natural" algorithms above. Theorem 3.1 rules out this possibility and demonstrates that both $\mathcal{A}_{\text{flip}}$ and $\mathcal{A}_{\text{obl}}$ are essentially optimal for $\mathcal{P}_n$.

To prove our theorem, we apply a standard adversary argument: We view the interaction between the learning algorithm $\mathcal{A}$ and the adversary $\mathcal{B}$ (which picks a target function from $\mathcal{P}_n$ and answers membership queries) as a game between two players. The adversary $\mathcal{B}$ responds to the membership queries in a way that forces $\mathcal{A}$ to ask many queries. It is convenient to think of $\mathcal{B}$ as if it does not choose a target concept in advance, but rather, at each moment, it has a class of potential target functions which are consistent with the queries made by $\mathcal{A}$ so far and the corresponding answers of $\mathcal{B}$. The learner's goal is to extract as much information as possible from each query; that is, to narrow down the class of potential target functions (or more precisely, to decrease the number of distinct minimal points of functions in the class). The game continues until all the consistent functions in the class have the same local minimum point. At this moment, and only at this moment, $\mathcal{A}$ can conclude what the minimal point is. (Note that this argument utilizes the fact that every function in $\mathcal{P}_n$ has a *unique* minimum point.)

In Section 3.1 we define the adversary's strategy, and in Section 3.2 we show that this strategy indeed causes the learning algorithm to ask many membership queries, as stated in Theorem 3.1. In Section 3.3 we informally present an alternative view of the lower bound proof.

3.1. The Adversary's Strategy

We assume, without loss of generality, that the algorithm $\mathcal{A}$ is "non-redundant;" that is, it never asks queries whose answers are implied by the answers to the previous queries. We first define this notion formally.

DEFINITION 3.1. Let $X = \{(x_1, \ell_1), (x_2, \ell_2), \ldots, (x_i, \ell_i)\}$ be a set of i labeled points in $\{0,1\}^n$; that is, $x_j \in \{0,1\}^n$ and $\ell_j \in \{0,1\}$ for $j = 1, \ldots, i$. We say that X is consistent with a function f if $f(x_1) = \ell_1, f(x_2) = \ell_2, \ldots, f(x_i) = \ell_i$. For a set $X \subseteq \{0,1\}^n$ of *unlabeled* points, we say that f is consistent with X if f is consistent with $\{(x, 0) : x \in X\}$.

[4] The modified algorithm iteratively takes a point w whose weight is i and it queries the value of f on the first $i-1$ sons of w. If it finds a son u such that $f(u) = 1$ it replaces w by u; otherwise it replaces w by its i-th son u without asking a query on u (it is possible that $f(u) = 0$). These iterations end when we reach $w = 0^n$; and in this stage the algorithm finds, using a binary search, a vertex on the path whose value is 1 and the value of its son on the path is 0. This vertex is a local minimum point.

DEFINITION 3.2. We say that a set $X = \{(x_1, \ell_1), (x_2, \ell_2), \ldots, (x_{i-1}, \ell_{i-1})\}$ of $i-1$ labeled points implies the value of a point x_i if there exists some $\ell_i \in \{0,1\}$ such that for every $f \in \mathcal{P}_n$ that is consistent with X it holds that $f(x_i) = \ell_i$.

We say that an algorithm $\mathcal{A}$ is *non-redundant* if for every $i \geq 1$ the following holds: Let $X_{i-1} = \{(x_1, \ell_1), (x_2, \ell_2), \ldots, (x_{i-1}, \ell_{i-1})\}$ be the set of first $i-1$ membership queries asked by $\mathcal{A}$ with their labels, and let x_i be its i membership query. Then, the value $f(x_i)$ is not implied by X_{i-1}.

After making the assumption that algorithm $\mathcal{A}$ is "non-redundant" we can consider a very simple strategy $\mathcal{B}$ for the adversary: it answers each query it gets by "0." [5] In the next lemma we prove that this strategy is valid; i.e., algorithm $\mathcal{A}$ when interacting with this adversary $\mathcal{B}$ finds a minimal point.

LEMMA 3.1. *For each $i \geq 1$, after $\mathcal{A}$ asks the i-th query and gets its label, there exists a function $f \in \mathcal{P}_n$ which is consistent with $\mathcal{B}$'s answers.*

Proof. Let $x_1, x_2, \ldots, x_i$ be $\mathcal{A}$'s first i queries while interacting with $\mathcal{B}$. The answer to x_i is not implied by the previous answers (since $\mathcal{A}$ asks the query x_i and $\mathcal{A}$ is non-redundant). In particular, there is some function $f \in \mathcal{P}_n$ that was consistent before the ith query (i.e., $f(x_1) = f(x_2) = \ldots = f(x_{i-1}) = 0$) and such that $f(x_i) = 0$. ■

The execution of algorithm $\mathcal{A}$ while interacting with adversary $\mathcal{B}$ is thus identical to the execution of $\mathcal{A}$ while getting answers according to some function $f \in \mathcal{P}_n$. By permuting the order of the bits we can assume, without loss of generality, that this function is f_{p_n} where $p_n \stackrel{\text{def}}{=} 1^n, 1^{n-1}0, \ldots, 110^{n-2}$. Therefore, $\mathcal{A}$ must output the minimal point $a_n \stackrel{\text{def}}{=} 110^{n-2}$. At the end of $\mathcal{A}$'s execution, there can be many functions consistent with $\mathcal{B}$'s answers; by the correctness of $\mathcal{A}$ all of these functions have the point a_n as their minimal point. That is, a_n is the only possible minimal point consistent with the negative answers of $\mathcal{B}$.

3.2. The Lower Bound Proof

Based on the above discussion, what $\mathcal{A}$ learns from the interaction with the specified adversary $\mathcal{B}$ is just a list of 0-points which indicate that a_n is the desired minimal point. Therefore, the question we are left with is the number of zero points one needs in order to prove that a_n is the only possible minimal point.

DEFINITION 3.3. A set $B_n \subseteq \{0,1\}^n$ is a *witness* for a_n being the minimal point if:

- The function f_{p_n} is consistent with B_n (i.e., $f_{p_n}(x) = 0$ for all $x \in B_n$), and
- The point a_n is the minimal point of every $f \in \mathcal{P}_n$ that is consistent with B_n.

The next lemma establishes a lower bound on the size of B_n and implies Theorem 3.1:

[5] As an intuition, consider the first query x_1. If the adversary would answer it with "1" then $\mathcal{A}$ can significantly restrict the set of possible minimal points (to those points in the second level whose set of 1s is a subset of the set of 1s in x_1). On the other hand, a "0" answer eliminates a certain set of paths (all those that go through x_1) but, unless x_1 itself is on the second level of the lattice, it does not eliminate even a single possible minimal point. However, when considering more than one query of $\mathcal{A}$ things are not as simple, and the main essence of the proof will be to show that this strategy is still good for the adversary.

LEMMA 3.2. *Let B_n be a witness that a_n is the minimal point. Then, $|B_n| \geq \binom{n}{2} - 1$ for every $n \geq 2$.*

Proof. We prove by induction on n that $|B_n| \geq \binom{n}{2} - 1$. The basis is trivial since $|B_2| \geq 0 = \binom{2}{2} - 1$.

For the induction step, we partition B_n into two disjoint sets B_n^0 and B_n^1 where $B_n^0 \stackrel{\text{def}}{=} B_n \cap \{0,1\}^{n-1} \times \{0\}$ and $B_n^1 \stackrel{\text{def}}{=} B_n \cap \{0,1\}^{n-1} \times \{1\}$. To give a lower bound on the size of B_n we give lower bounds on the size of both B_n^0 and B_n^1. The first observation is that B_n^0 "nearly" satisfies the induction hypothesis for $n-1$. That is,

Claim. $|B_n^0| \geq \binom{n-1}{2} - 1$ for every $n \geq 3$.

Proof. Define $\hat{B}_n^0 \stackrel{\text{def}}{=} \left\{\hat{b} \in \{0,1\}^{n-1} : \hat{b} \circ 0 \in B_n^0\right\}$. We argue that the set $\hat{B}_n^0$ is a witness that a_{n-1} is a minimal point: First, $f_{p_{n-1}}$ is consistent with $\hat{B}_n^0$ (since f_{p_n} is consistent with B_n). Second, let $\hat{f} \in \mathcal{P}_{n-1}$ be any function that is consistent with $\hat{B}_n^0$; consider the function $f \in \mathcal{P}_n$ where $f(1^n) = 1$, and for every $\hat{x} \in \{0,1\}^{n-1}$ it holds that $f(\hat{x} \circ 0) = \hat{f}(\hat{x})$ and if $\hat{x} \neq 1^{n-1}$ then $f(\hat{x} \circ 1) = 0$. Thus, f is consistent with B_n (since $\hat{f}$ is consistent with $\hat{B}_n^0$). Therefore, the minimal point of f is a_n, which implies that the minimal point of the function $\hat{f}$ is a_{n-1}. Thus, $\hat{B}_n^0$ satisfies the induction hypothesis for $n-1$. Hence, $|B_n^0| = |\hat{B}_n^0| \geq \binom{n-1}{2} - 1$. ■

Next we argue that B_n^1 is not too small either:

Claim. $|B_n^1| \geq n-1$ for every $n \geq 3$.

Proof. Consider the functions $f \in \mathcal{P}_n$ which correspond to paths in the lattice $\{0,1\}^{n-1} \times \{1\}$; that is, every node in the path is in $\{0,1\}^{n-1} \times \{1\}$. The set B_n^1 guarantees that any such function is not consistent with B_n (since the points in B_n^0 are consistent with every path in $\{0,1\}^{n-1} \times \{1\}$). Furthermore, $1^n \notin B_n^1$ since B_n is consistent with f_{p_n}.

We prove, by induction on n, that every set $C_n \subseteq \{0,1\}^{n-1} \times \{1\} \setminus \{1^n\}$ that is not consistent with every f in the lattice $\{0,1\}^{n-1} \times \{1\}$ (that is, C_n contains a point on every such path f and does not contain the point 1^n) has cardinality at least $n-1$. For the induction basis notice that $\{011, 101\} \subseteq C_3$, thus $|C_3| \geq 2$. For the induction step, if $\left\{1^{n-2}01, 1^{n-3}011, \ldots, 01^{n-1}\right\} \subseteq C_n$ then the claim follows. Otherwise, by permuting the order of indices, assume that $1^{n-2}01 \notin C_n$. Partition C_n into two disjoint sets C_n^0 and C_n^1 where $C_n^0 \stackrel{\text{def}}{=} C_n \cap \{0,1\}^{n-2} \times \{01\}$ and $C_n^1 \stackrel{\text{def}}{=} C_n \cap \{0,1\}^{n-2} \times \{11\}$. First notice that $|C_n^1| \geq 1$ since otherwise the function corresponding to the path $1^n, 01^{n-1}, \ldots, 0^{n-2}11$ would have been consistent with C_n. Second, define the set $\hat{C}_n^0 \stackrel{\text{def}}{=} \left\{\hat{c} \circ 1 \in \{0,1\}^{n-1} : \hat{c} \circ 01 \in C_n^0\right\}$. The set $\hat{C}_n^0$ satisfies the induction hypothesis for $n-1$, hence, $|C_n^0| = |\hat{C}_n^0| \geq n-2$ and $|C_n| = |C_n^0| + |C_n^1| \geq n-1$. ■

To complete the proof of Lemma 3.2 (and Theorem 3.1), note that

$$|B_n| = |B_n^0| + |B_n^1| \geq \binom{n-1}{2} - 1 + n - 1 = \binom{n}{2} - 1.$$

■

3.3. An Alternative View of the Lower Bound Proof

The above lower bound proof uses two nested inductions (in the proofs of Lemma 3.2 and its second claim). To explain the intuition behind the proof we "unfold" these inductions. The basic idea of the lower bound proof is that, initially, the set of candidate minimal points is "large." Assuming that the adversary behaves according to the strategy specified above, we show that each query made by the learning algorithm reduces the number of candidate minimal points by at most 1. As a consequence, we get that at least $\binom{n}{2} - 1$ membership queries are needed in order to eliminate all but one minimal point.

More precisely, the proof proceeds as follows. First, we have argued that there exists a function f consistent with all answers given by the adversary $\mathcal{B}$ during the execution. Without loss of generality, we can assume that f_{p_n} is a consistent function. We use the path p_n to build a tree (a subgraph of the lattice graph G_B) which possesses the following two properties:

1. The path p_n is the tree's "trunk": each vertex v of p_n gives rise to a set of paths which "grow" towards the second level.

2. All the paths in the tree are "almost" *vertex disjoint* and cover the second level entirely. More formally, the paths' lower ends are all the points of the second level; moreover, the lower ends of all the paths are distinct; and, the only vertices that the paths may share are those on the "trunk" p_n.

Such a tree is described in Fig. 1. This tree is implicitly built in Lemma 3.2. Specifically,

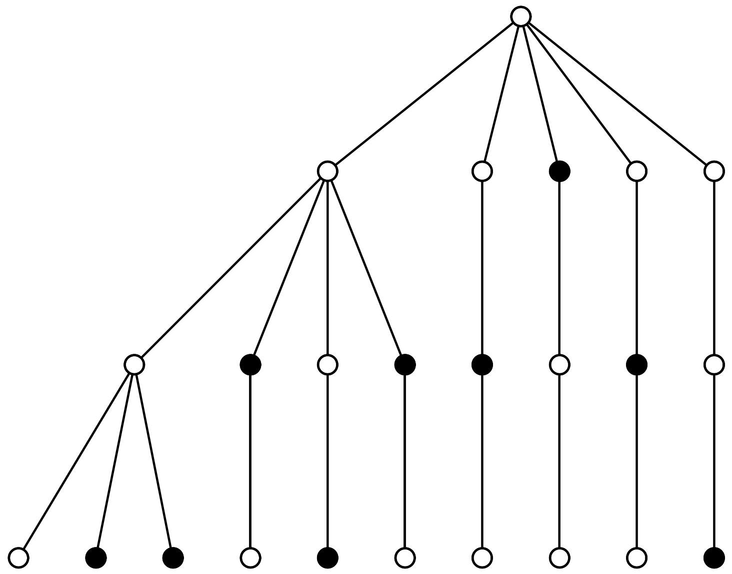

FIG. 1. The "almost" disjoint paths: The path p_n consists of the leftmost vertex in each level; the blackened points correspond to the queries of $\mathcal{A}$.

in each application of the induction step of the second claim in Lemma 3.2 we add a new path to the tree (when we claim that $|C_n^1| \geq 1$).

Since at the end only one point remains in the set of candidate minimal points, it means that the learning algorithm $\mathcal{A}$ was compelled to reject all the paths of the tree except p_n (this follows from the fact that no two paths share the common lower endpoint). The "almost"-disjointness of the paths ensures that $\mathcal{A}$ asks a separate membership query for each path in order to eliminate it (the points on the trunk are all positive; thus, by the definition of $\mathcal{B}$, the algorithm $\mathcal{A}$ does not query on any of them). As the number of paths is equal to the number of points on the second level, that is $\binom{n}{2}$, this establishes the desired lower bound. (A full proof along these lines can be found in the conference version of this paper.)

4. THE LOWER BOUND FOR RANDOMIZED ALGORITHMS

In this section we show that randomization cannot substantially improve the performance of local minimum finding algorithms. We prove that the expected number of queries asked by every randomized algorithm for this task is at least half of the lower bound of the deterministic case. In other words, $\Omega(n^2)$ queries are required even if we use randomized algorithms.

First, we define the notion of "randomized algorithms for finding local minimum points." Roughly speaking, these are algorithms which always succeed in finding a local minimum point, but for which the *expected* search time is considered. More formally, a randomized algorithm $\mathcal{A}$ produces local minimum points for the class $\mathcal{F}$ if for every $f \in \mathcal{F}$ and every input u_0 such that $f(u_0) = 1$ the algorithm $\mathcal{A}$, given an access to a membership oracle for f, always halts and outputs a local minimum point u of f such that $u \leq u_0$. The expected membership query complexity of $\mathcal{A}$ on $\mathcal{F}$, denoted RMC$(\mathcal{A}, \mathcal{F})$, is the maximum over all $f \in \mathcal{F}$ and u_0 such that $f(u_0) = 1$ of the *expected* number of queries that $\mathcal{A}$ makes on input u_0 (using a membership oracle for f).

Next, we present a randomized algorithm in which the number of queries is approximately $n^2/4$, i.e., half the lower bound for deterministic algorithms. We consider a randomized implementation of the search procedure of Bshouty, which iteratively takes a point x such that $f(x) = 1$ and finds a point y among the sons of x such that $f(y) = 1$. The randomized algorithm scans the sons of x in a random order. If x is not a local minimum then the expected number of queries of the randomized algorithm for a point x of weight i is at most $(i+1)/2$ while if x is a local minimum then the number of queries is at most i (but this happens only once). Therefore, the expected number of queries of all weights is at most $n + \sum_{i=1}^{n} \frac{i+1}{2} \leq n + \frac{1}{2}\binom{n+2}{2} = \frac{n^2}{4} + O(n)$.

The idea behind the lower bound for the randomized case is that the adversary first chooses the target function at random from $\mathcal{P}_n$ and answers the queries according to this function. We show that if the randomized algorithm goes down the lattice from father to son (as is the case in the search procedures of [3, 13]), then the expected number of queries needed to find the son in iteration i is $\Omega(i)$, and therefore the algorithm uses $\Omega(n^2)$ queries. However, if the algorithm "tries to skip a level" then this alone causes him to make $\Omega(n^2)$ queries. That is, if the algorithm does not know a point x of weight at most $i > n/2$ such that $f(x) = 1$ then the probability that any query y of weight less than i is a positive point, i.e., $f(y) = 1$, is $O(1/n^2)$, and the expected number of queries the algorithm makes until it finds a positive point of weight less than i is $\Omega(n^2)$.

In Theorem A.1 we prove an asymptotically tight lower bound of $n^2/4 - o(n^2)$ membership queries for randomized algorithms for finding local minimum. For the clarity of presentation, we first prove in Theorem 4.1 a lower bound of $\Omega(n^2)$ with a smaller leading constant. We defer the more technical proof of the asymptotically tight lower bound to the appendix.

THEOREM 4.1. *For any* randomized *algorithm $\mathcal{A}$ and any concept class $\mathcal{F}$, such that $\mathcal{P}_n \subseteq \mathcal{F}$,*

$$\mathrm{RMC}(\mathcal{A}, \mathcal{F}) = \Omega(n^2).$$

Proof. Let $\mathcal{A}$ be any randomized algorithm that finds local minimum points for the functions in $\mathcal{P}_n$. As in the deterministic case, we view the interaction between the learning algorithm $\mathcal{A}$ and the adversary $\mathcal{R}$ (a randomized algorithm which picks a target function f and a starting point u, and answers membership queries) as a game between two players. The strategy of the adversary is simple, at the beginning of the game it picks at random, with uniform distribution, a function f from $\mathcal{P}_n$, it picks $u = 1^n$ as a starting point, and it answers all membership queries according to f. If the expected number of queries asked by the learning algorithm in this game is $\Omega(n^2)$ then there exists some function f such that the expected number of queries of the learning algorithm $\mathcal{A}$ when f is the target function is $\Omega(n^2)$ (in fact, this is true for at least a constant fraction of the functions in $\mathcal{P}_n$).

We assume that the last query the algorithm makes is on the local minimum point it outputs (this adds at most one query to the algorithm). Since $\mathcal{A}$ never fails, the answer to this query is 1. Let S_i be the expected number of queries of weight i that the algorithm makes (all expectations are over the random choice of $f \in \mathcal{P}_n$ and the random choices of the learning algorithm $\mathcal{A}$). Assume, for sake of contradiction, that the expected number of queries of algorithm $\mathcal{A}$ against adversary $\mathcal{R}$ is less than $\frac{1}{16}\binom{n}{2}$. Let i, where $2 \leq i \leq n-1$, be the maximum index such that $S_i < (i+1)/16$ (by the above assumption such an index exists). Since

$$\sum_{j=2}^{n-1} S_j < \frac{1}{16}\binom{n}{2},$$

and for every j, where $i+1 \leq j \leq n-1$, it holds that $S_j \geq (j+1)/16$, we have

$$\sum_{j=2}^{i} S_j = \sum_{j=2}^{n-1} S_j - \sum_{j=i+1}^{n-1} S_j < \frac{i^2 + 3i + 2 - 2n}{32} \leq \frac{i^2 + i}{32}.$$

We construct an algorithm, $\mathcal{A}_1$, which runs algorithm $\mathcal{A}$ as long as algorithm $\mathcal{A}$ uses less than $(i^2+i)/8$ queries of weight at most i, and less than $(i+1)/4$ queries of weight exactly i. If at least one of these conditions is violated then algorithm $\mathcal{A}_1$ terminates without finding a local minimum point. By Markov inequality, algorithm $\mathcal{A}_1$ finds a correct local minimum point with probability at least $1/2$ (again, the probability is over the uniform distribution of $f \in \mathcal{P}_n$ and over the random choices of $\mathcal{A}_1$).

We use Yao's principle [29] to construct a *deterministic* algorithm $\mathcal{A}_2$ that succeeds for at least $1/2$ of the functions in $\mathcal{P}_n$ and still has the same properties as $\mathcal{A}_1$: it uses less than $(i^2+i)/8$ queries of weight at most i and less than $(i+1)/4$ queries of weight exactly i. (By an averaging argument, there must be a random input r such that $\mathcal{A}_1$ with random input r finds the local minimum point for at least half the functions in $\mathcal{P}_n$; the deterministic algorithm $\mathcal{A}_2$ is obtained by running $\mathcal{A}_1$ with the fixed random string r.) This implies that there is some path p' from 1^n to a vertex v of weight $i+1$ such that for at least $1/2$ of the paths p'' from sons of v to a vertex in the second level, algorithm $\mathcal{A}_2$ finds the local minimum point of $p'p''$. Let $\mathcal{P}'_n$ be the set of paths $p'p''$ where p'' is a path from v to a vertex in the second level. Note that the number of paths that pass through each of the $\binom{i+1}{j}$ descendants of v of weight j is the same. Thus,

PROPOSITION 4.1. *Let w be a descendant of v such that* $\text{weight}(w) = j$. *The fraction of paths in $\mathcal{P}'_n$ that pass through w is $1/\binom{i+1}{j}$.*

We consider only functions in $\mathcal{P}'_n$ for which the answers of the membership oracle is zero for every query of $\mathcal{A}_2$ on points of weight at most i. For all these functions, $\mathcal{A}_2$ asks exactly the same queries (since the query that the *deterministic* $\mathcal{A}_2$ makes in some step depends only on the answers it got for previous queries). We say that a query w excludes a path p if w is on the path p. By Proposition 4.1, every query on a point of weight i excludes at most a fraction of $1/(i+1)$ of the paths in $\mathcal{P}'_n$, and there are less than $(i+1)/4$ such queries. Similarly, every query of weight $j < i$ excludes at most a fraction of $\frac{2}{i(i+1)}$ (since $2 \leq j \leq i-1$), and there are less than $(i^2+i)/8$ such queries. All together the fraction of paths that are excluded is less than half.

Algorithm $\mathcal{A}_2$ fails for all functions in $\mathcal{P}'_n$ for which it gets only negative answers to queries of weight at most i (by the assumption that the last query of algorithm $\mathcal{A}$ is the local minimum point it outputs). Algorithm $\mathcal{A}_2$, with an oracle to the function corresponding to the path $p'p''$, gets a positive answer to a query of weight at most i only if it excludes $p'p''$. That is, $\mathcal{A}_2$ succeeds on less than half of the functions in $\mathcal{P}'_n$. This implies that our assumption, that the expected number of queries of algorithm $\mathcal{A}$ is less than $\frac{1}{16}\binom{n}{2}$ for every function $f \in \mathcal{P}_n$, is false. This completes the proof of Theorem 4.1. ■

5. FINDING LOCAL MINIMA OF t-TERM DNF FUNCTIONS

5.1. Upper Bound

In this section we show that finding a minimal point of a t-term DNF function requires only $\Theta(tn)$ membership queries. Our procedure FIND_MIN, described in Fig. 2, repeats Angluin's procedure (described in the Introduction) until a local minimum point is found. We show that this procedure terminates after $O(t)$ iterations. This procedure is an implementation of Bshouty's procedure (also described in the Introduction) where we specify how to search for the next index that should be flipped: in each iteration go over *all* the variables and try to flip them; if you succeed in flipping the i-th bit then proceed the search with the modified w from index $i+1$. We emphasize that there are other strategies for implementing Bshouty's procedure that require $\Theta(n^2)$ queries even for 1-term DNF formulae (e.g., after finding an index start a new search from $i \leftarrow 1$ to n).

The procedure FIND_MIN is detailed in Fig. 2. As before, $w[i]$ denotes the i-th bit of w and $\text{flip}(w,i)$ is an operator which is given a string $w \in \{0,1\}^n$ such that $w[i]=1$, and flips the i-th bit from 1 to 0.

To understand why $O(t)$ iterations suffice, we return to the monotone case. In this case, once the algorithm finds that $f(\text{flip}(w,i)) = 0$ then from the monotonicity $f(\text{flip}(u,i)) = 0$ for every $u \leq w$. Thus, the algorithm does not need to check the i-th bit again. In the non-monotone case, however, $\text{flip}(u,i)$ might satisfy a term of f which is not satisfied by $\text{flip}(w,i)$. Our observation is that this implies that a term that was not satisfied by w itself

ALGORITHM FIND_MIN(w)

REPEAT:
    change $\leftarrow$ FALSE.
    FOR $1 \leq i \leq n$:
        IF $w[i] = 1$ and $f(\text{flip}(w, i)) = 1$ THEN
            $w \leftarrow \text{flip}(w, i)$.
            change $\leftarrow$ TRUE.
UNTIL change = FALSE.
The last value of w is a minimal point.

FIG. 2. An algorithm for finding a local minimum point.

is now satisfied by flip(u, i). Furthermore, we prove that if a term is satisfied by w and later is not satisfied by the modified w then it would not be satisfied by any modified w in the future. All together, we will prove that if the procedure has not terminated in some iteration then a new term has to be satisfied and every term can be new only once, thus after $O(t)$ iterations the procedure terminates.

LEMMA 5.1. *Let T be a term, and $v, y \in \{0,1\}^n$ be assignments such that $v \leq y$. If $T(y) = 1$ while $T(v) = 0$ then $T(u) = 0$ for every $u \leq v$.*

Proof. Since $T(y) = 1$ then for every negated variable $\overline{x}_i$ in the term T, the corresponding bit $y[i]$ equals 0. Since $v \leq y$, these bits are also set to zero in v. Thus, the fact that $T(v) = 0$ implies that for some i the variable x_i appears in T while $v[i] = 0$. Since $u \leq v$, we have $u[i] = 0$ and so u does not satisfy the term T as well. ■

LEMMA 5.2. *Let $f = \vee_{i=1}^{t} T_i$ for some terms $T_1, \ldots, T_t$, and let $u, v \in \{0,1\}^n$ be assignments such that $u \leq v$ and $u[i] = 1$. If $f(\text{flip}(u, i)) = 1$ while $f(\text{flip}(v, i)) = 0$ then there is some term T_j such that $T_j(\text{flip}(u, i)) = 1$ while $T_j(y) = 0$ for every $y \geq v$.*

Proof. Since $f(\text{flip}(u, i)) = 1$, there is a j such that $T_j(\text{flip}(u, i)) = 1$. Notice that

$$\text{flip}(u, i) \leq \text{flip}(v, i) \leq v \leq y.$$

But $T_j(\text{flip}(v, i)) = 0$ (since $f(\text{flip}(v, i)) = 0$), thus, by Lemma 5.1, $T_j(y) = 0$. ■

THEOREM 5.1. *Let $t \leq n/2$ and $f = \vee_{i=1}^{t} T_i$ for some terms $T_1, \ldots, T_t$. Algorithm* FIND_MIN, *described in Fig. 2, finds a local minimum point of f asking at most $(2t - 1)(n - t) + n = O(tn)$ membership queries (provided that its input is a point w such that $f(w) = 1$).*

Proof. Denote the number of iterations of the algorithm by k. In every iteration of the algorithm, except the last iteration, there exists at least one index i such that $w[i]$ is flipped from 1 to 0. For every iteration j, fix such an index and denote it by i_j. Furthermore, let $\mathcal{T}_j$ be the set of terms satisfied by w after bit i_j was flipped in the j-th iteration. For completeness, let $\mathcal{T}_0$ be the set of terms satisfied at the beginning of the execution of the algorithm.

Next we use Lemma 5.2 to show that for every j, where $2 \leq j \leq k-1$, there is a term $T_{\ell_j} \in \mathcal{T}_j \setminus \mathcal{T}_{j-2}$. To apply Lemma 5.2, let y be the value of w in the algorithm after the i_{j-2} bit is flipped, v be the value of w when the i_j-th bit is checked in iteration $j-1$, and u be the value of w before the i_j-th bit is flipped in iteration j. Since bit i_j is flipped in iteration j and not in iteration $j-1$ then $f(\text{flip}(u,i)) = 1$ while $f(\text{flip}(v,i)) = 0$, and Lemma 5.2 implies the existence of the term $T_{\ell_j} \in \mathcal{T}_j \setminus \mathcal{T}_{j-2}$.

To complete the proof let T_{i_0} be any term that satisfies w at the beginning of the execution of the algorithm, and consider the sequence

$$T_{i_0}, T_{i_2}, \ldots, T_{i_{2\cdot\lfloor\frac{k-1}{2}\rfloor}}.$$

By Lemma 5.1, each term of f appears at most once in this sequence of length $\lfloor\frac{k-1}{2}\rfloor + 1$, thus the length of the sequence is at most t. Hence, the number of iterations k is at most $2t$. ∎

Our analysis of the complexity of finding local minimum points of t-term DNF improves the time and membership query complexity of learning the class t-term DNF by a factor of n/t. Among the many algorithms for learning this class for $t = O(\log n)$ [12, 13, 14, 21, 9], the most efficient one is the algorithm of Bshouty [13] based on his monotone theory. The number of equivalence queries in this algorithm for learning a t-term DNF function is $O(u \cdot t)$ where u is the size of an (n,t) universal set. In addition, the algorithm calls $O(u \cdot t)$ times the procedure FIND_TERM, and no membership queries are used otherwise. The size of the smallest deterministic construction of an (n,t) universal set known to date [25] is $2^t \cdot t^{O(\log t)} \cdot \log n$, and the smallest known randomized construction has size $O(t \cdot 2^t \cdot \log n)$. With our implementation of FIND_TERM, we get a deterministic algorithm which uses $O(2^t \cdot t^{O(\log t)} \cdot n \log n)$ membership queries, and a randomized algorithm which uses $O(t^2 \cdot 2^t \cdot n \log n)$ membership queries.

Bshouty [14] presented an alternative algorithm for this class based on the divide and conquer approach. This algorithm, whose running time and query complexity are inferior to the [13] algorithm, calls $O(t \cdot 2^t \cdot n)$ times the procedure FIND_TERM. In this case we improve the membership query complexity by a factor of n (and not only n/t), because most calls to FIND_TERM are with functions with less than t terms: for every d, where $1 \leq d \leq t$, there are $O(t \cdot 2^d \cdot n)$ calls with a $(t-d)$-term DNF function. Thus, all together, the number of membership queries in this deterministic algorithm is

$$O\left(\sum_{d=1}^{t} n \cdot t \cdot 2^d \cdot n(t-d)\right) = O(t \cdot 2^t \cdot n^2).$$

5.2. Lower Bound for the Class t-term DNF

We prove a matching lower bound of $\Omega(tn)$ for finding a minimal point of a function taken from the class t-term DNF ($t \leq n$). This lower bound can be proven by modifying the proof of Theorem 3.1. We give an alternative proof which uses Theorem 3.1 (without modifying its proof). We first need some notation. For $1 \leq t \leq n-1$, the class $\mathcal{P}_{n,t}$

consists of all boolean functions which correspond to paths starting at the top vertex of the lattice (i.e., 1^n) and ending at a vertex on the $(n-t+1)$ level (that is, each path contains t vertices):

$$\mathcal{P}_{n,t} = \{\ f_p \mid p = \{v_0, v_1, ..., v_{t-1}\} \text{ is a path in } G_B\,,\ \text{ and } v_0 = 1^n\,\}\,.$$

(E.g., the class $\mathcal{P}_n$, defined in Section 2, is the same class as $\mathcal{P}_{n,n-1}$; also note that weight$(v_{t-1}) = n-t+1$.) We will show in Corollary 5.1 that every function in $\mathcal{P}_{n,2t}$ has a t-term DNF representation (it is easy to see that such a DNF representation exists for every function in $\mathcal{P}_{n,t}$). Therefore, the following theorem implies the lower bound for t-term DNF.

THEOREM 5.2. *Let t and n be integers such that $1 \le t \le (n-1)/2$. For any deterministic algorithm $\mathcal{A}$ and any concept class $\mathcal{F}$, such that $\mathcal{P}_{n,2t} \subseteq \mathcal{F}$,*

$$\mathrm{MC}(\mathcal{A}, \mathcal{F}) \ge (2t-1)(n-t) - 1 = \Omega(tn).$$

Proof. The proof is by a reduction to Theorem 3.1. Given an algorithm $\mathcal{A}$ which finds a local minimum point for every function in $\mathcal{P}_{n,2t}$, we construct an algorithm $\mathcal{A}'$ which finds a local minimum point for every function in $\mathcal{P}_n$ and uses at most $\mathrm{MC}(\mathcal{A}, \mathcal{F}) + \binom{n-2t+1}{2}$ membership queries. Thus, by Theorem 3.1,

$$\mathrm{MC}(\mathcal{A}, \mathcal{F}) + \binom{n-2t+1}{2} \ge \mathrm{MC}(\mathcal{A}', \mathcal{P}_n) \ge \binom{n}{2} - 1.$$

That is, $\mathrm{MC}(\mathcal{A}, \mathcal{F}) \ge (2t-1)(n-t) - 1$.

Algorithm $\mathcal{A}'$ works as follows. Given a function $f \in \mathcal{P}_n$, algorithm $\mathcal{A}'$ first executes algorithm $\mathcal{A}$ in order to find the length $2t$ prefix of the path. For this, if $\mathcal{A}$ asks a membership query on an assignment whose weight is at most $n-2t$ algorithm $\mathcal{A}'$ answers it 0 (without actually asking a membership query) and for every other membership query $\mathcal{A}'$ answers using a membership query to f. Thus, $\mathcal{A}$ sees a function from $\mathcal{P}_{n,2t}$ and outputs a point w whose weight is $n-2t+1$ such that $f(w) = 1$ (note that for every function in $\mathcal{P}_{n,2t}$ there is a unique such w). Next, algorithm $\mathcal{A}'$ queries all descendants of weight 2 of w, the output of algorithm $\mathcal{A}$. This requires additional $\binom{n-2t+1}{2}$ queries. This concludes the description of $\mathcal{A}'$, whose complexity is at most $\mathrm{MC}(\mathcal{A}, \mathcal{F}) + \binom{n-2t+1}{2}$, and completes the proof of the theorem. ■

COROLLARY 5.1. *Let t and n be integers such that $1 \le t \le (n-1)/2$. For any deterministic algorithm $\mathcal{A}$, and any concept class $\mathcal{F}$ that contains the class t-term DNF,*

$$\mathrm{MC}(\mathcal{A}, \mathcal{F}) \ge (2t-1)(n-t) - 1 = \Omega(tn).$$

Proof. We will show that every function $f_p \in \mathcal{P}_{n,2t}$ can be represented by a t-term DNF formula. It is enough to show this for the path

$$p = \left\{1^n, 01^{n-1}, \ldots, 0^{2t-1}1^{n-2t+1}\right\}.$$

(This is not a restriction since we can get p from any path in $\mathcal{P}_{n,2t}$ by appropriately permuting the order of the coordinates $\{1,\ldots,n\}$.)

We next describe the terms of the formula. For every i, where $0 \le i \le t-1$, define the term

$$T_i = \left(\bigwedge_{j=1}^{2i} \overline{x}_j\right) \wedge \left(\bigwedge_{j=2i+2}^{n} x_j\right).$$

(Notice that the variable x_{2i+1} and its negation do not appear in T_i). There are exactly two assignments that satisfy T_i, namely $0^{2i}1^{n-2i}$ and $0^{2i+1}1^{n-2i-1}$. Thus, the formula

$$T_1 \vee T_2 \vee \ldots \vee T_t$$

computes the function f_p. ∎

We next prove that the $\Omega(tn)$ lower bound for t-term DNF holds for randomized algorithms as well. The proof of the lower bound for deterministic algorithms for t-term DNF relies on the fact that the lower bound we prove for deterministic algorithms for $\mathcal{P}_n$ is tight. As the lower bound for randomized algorithms is not tight (there is an additive difference of $o(n^2)$), we need to present a different reduction. For this reduction we define the class of functions $\widehat{\mathcal{P}}_{n,t}$ which contains the functions corresponding to paths of length at most $t+1$ starting from an arbitrary vertex.

$$\widehat{\mathcal{P}}_{n,t} = \{\ f_p\ \mid p = \{v_0, v_1, \ldots, v_k\} \text{ is a path in } G_B,\ \text{and}\ \ k \le t\,\}\,.$$

Clearly, the class $\widehat{\mathcal{P}}_{n,t}$ is contained in the class t-term DNF.

THEOREM 5.3. *Let t and n be integers such that* $1 \le t \le n-2$. *For any randomized algorithm* $\mathcal{A}$, *and any concept class* $\mathcal{F}$ *that contains the class* $\widehat{\mathcal{P}}_{n,t}$,

$$\mathrm{RMC}(\mathcal{A}, \mathcal{F}) = \Omega(tn).$$

Proof. The proof is by a reduction to the lower bound proved in Theorem A.1. Given an algorithm $\mathcal{A}$ which finds a local minimum point for every function in $\widehat{\mathcal{P}}_{n,t}$, we will construct an algorithm $\widehat{\mathcal{A}}$ which finds a local minimum point for every function in $\mathcal{P}_{n,n-2}$ by calling $\mathcal{A}$ at most $(n/t+1)$ times. Thus, by Theorem A.1,

$$\left(\frac{n}{t}+1\right) \cdot \mathrm{RMC}(\mathcal{A}, \mathcal{F}) \ge \mathrm{RMC}(\widehat{\mathcal{A}}, \mathcal{P}_n) \ge \frac{n^2}{4} - o(n^2).$$

That is, $\mathrm{RMC}(\mathcal{A}, \mathcal{F}) \ge \Omega(tn)$.

Algorithm $\widehat{\mathcal{A}}$ works as follows. Given a function $f \in \mathcal{P}_n$, algorithm $\widehat{\mathcal{A}}$ iteratively uses algorithm $\mathcal{A}$ to find positive points of decreasing weights. That is, before iteration i, the algorithm has a point x of weight $n-it$ such that $f(x)=1$, and it finds a point y of weight $\max\{\,n-(i+1)t,\ 3\,\}$ such that $f(y)=1$. For this, if $\mathcal{A}$ asks a membership query on an assignment whose weight is more than $n-it$ or less than $n-(i+1)t$ algorithm $\widehat{\mathcal{A}}$ answers it 0 (without actually asking a membership query) and for every other membership query $\widehat{\mathcal{A}}$ answers using a membership query to f. Thus, $\mathcal{A}$ sees a function from $\widehat{\mathcal{P}}_{n,t}$ and outputs a point

y such that $\text{weight}(y) = \max\{\text{weight}(x) - t,\ 3\}$ and $f(y) = 1$. After $\lceil \frac{n}{t} \rceil$ iterations, algorithm $\widehat{\mathcal{A}}$ finds a positive point y of weight 3, which is the local minimum of f. The query complexity of $\widehat{\mathcal{A}}$ is at most $\left(\frac{n}{t} + 1\right) \cdot \text{RMC}(\mathcal{A}, \mathcal{F})$, which completes the proof of the theorem. ■

APPENDIX A

Asymptotically Tight Lower Bound for Randomized Algorithms

In this section we prove a lower bound of $n^2/4 - o(n^2)$ for randomized algorithms (i.e., we improve the leading constant of the lower bound proved in Theorem 4.1). This lower bound is asymptotically tight since there is a randomized algorithms for finding local minimum points that asks $n^2/4 + O(n)$ queries (see Section 4).

The first step in the improved lower bound is that the adversary $\mathcal{R}$ chooses paths from 1^n to the third level; that is, paths from $\mathcal{P}_{n,n-2}$ (instead of $\mathcal{P}_n$). Recall that S_i denotes the expected number of queries asked by $\mathcal{A}$ on points of weight i (where this time the expectation is over a random choice of a path from $\mathcal{P}_{n,n-2}$ and, as before, the random choices of $\mathcal{A}$); without loss of generality $S_i = 0$ for all $i < 3$. Assume, for sake of contradiction, that the expected number of queries asked by algorithm $\mathcal{A}$ against adversary $\mathcal{R}$ is at most $n^2/4 - 8 \cdot n^{2/3}$. Then,

LEMMA A.1. *If* $\text{RMC}(\mathcal{A}, \mathcal{F}) < \frac{n^2}{4} - 8 \cdot n^{3/2}$ *then there is a level* i*, where* $3 \le i \le n-1$*, such that*

$$S_i \le \frac{i+1}{2}\left(1 - \frac{16}{\sqrt{i+1}}\right) \text{ and } S_{i-1} \le i. \tag{A.1}$$

Proof. Assume that the claim is false. That is, no i satisfies (A.1). Let

$$A \stackrel{\text{def}}{=} \left\{ i \ :\ 3 \le i \le n-1,\ S_i \le \frac{i+1}{2}\left(1 - \frac{16}{\sqrt{i+1}}\right) \right\}$$

and

$$B \stackrel{\text{def}}{=} \{ i \ :\ 3 \le i \le n-2,\ i \notin A, \text{ and } i+1 \notin A \}.$$

If $i \in B$ then $S_i > \frac{i+1}{2}\left(1 - \frac{16}{\sqrt{i+1}}\right)$ (since $i \notin A$). Moreover, since i does not satisfy (A.1), if $i \in A$ then

$$S_i + S_{i-1} \ \ge\ S_{i-1} \ >\ i \ =\ \frac{i+1}{2} + \frac{i-1}{2}.$$

This implies that if $i \in A$ then $i-1 \notin A$ (and obviously $i-1 \notin B$). We next bound the expected number of queries that $\mathcal{A}$ asks:

$$\begin{aligned} \sum_{i=3}^{n-1} S_i \ &=\ \sum_{i \in B} S_i + \sum_{i \in A} (S_{i-1} + S_i) \ >\ \sum_{i=3}^{n-1} \frac{i+1}{2}\left(1 - \frac{16}{\sqrt{i+1}}\right) \\ &\ge\ \frac{(n+4)(n-3)}{4} - 8 \cdot n^{3/2} \ \ge\ \frac{n^2}{4} - 8 \cdot n^{3/2}. \end{aligned}$$

This contradicts the assumption of the lemma. ■

For the rest of the proof fix the maximum level i, where $3 \le i \le n-1$, that satisfies (A.1). Similarly to the proof of Lemma A.1 we get

$$\begin{aligned}\sum_{k=i+1}^{n-1} S_k &\ge \sum_{k=i+1}^{n-1} \frac{i+1}{2}\left(1-\frac{16}{\sqrt{i+1}}\right)\\ &\ge \frac{(n+i+2)(n-i-1)}{4} - 8(n-i)\cdot n^{1/2}\\ &\ge \frac{n^2}{4} - 8\cdot n^{3/2} - \frac{i^2-i}{4}. \qquad \text{(A.2)}\end{aligned}$$

Thus, by the assumption that the expected number of queries that $\mathcal{A}$ asks is small and by Inequality (A.1):

$$\sum_{k=3}^{i} S_k = \sum_{k=3}^{n-1} S_k - \sum_{k=i+1}^{n-1} S_k \le \frac{i^2-i}{4}. \qquad \text{(A.3)}$$

Let C_i be a random variable denoting the number of queries asked by algorithm $\mathcal{A}$ against adversary $\mathcal{R}$ on level i. Using this notation $S_i = \mathrm{E}(C_i)$. The next lemma is the main tool for proving the tighter lower bound; it replaces the somewhat crude usage of Markov Inequality in the proof of Theorem 4.1 by a more refined analysis.

LEMMA A.2. *For every integer $j \ge 0$,*

$$\Pr[C_i \le j] \le \frac{j}{i+1} + \frac{8}{\sqrt{i+1}},$$

where the probability is over the uniform distribution on the paths in $\mathcal{P}_{n,n-2}$ and over the random choices of $\mathcal{A}$.

Proof. Assume that $\Pr[C_i \le j] > \frac{j}{i+1} + \frac{8}{\sqrt{i+1}}$ for some j. By (A.1), $\mathrm{E}(C_{i-1}) = S_{i-1} \le i$, thus, by Markov Inequality,

$$\Pr\left[\, C_{i-1} > \frac{\sqrt{i+1}}{4}\cdot i \,\right] \le \frac{4}{\sqrt{i+1}}.$$

By (A.3), linearity of expectation (i.e., $\mathrm{E}(\sum_{k=3}^{i} C_k) = \sum_{k=3}^{i} S_k$), and Markov Inequality

$$\Pr\left[\sum_{k=3}^{i-2} C_k > \frac{(i-1)i\sqrt{i+1}}{12}\right] \le \Pr\left[\sum_{k=3}^{i} C_k > \frac{\sqrt{i+1}}{3}\cdot\frac{i^2-i}{4}\right] \le \frac{3}{\sqrt{i+1}}.$$

As in the proof of Theorem 4.1, we construct a deterministic algorithm $\mathcal{A}_2$ which makes at most j queries of weight i, at most $\frac{i\sqrt{i+1}}{4}$ queries of weight $i-1$, at most $\frac{(i-1)i\sqrt{i+1}}{12}$ queries of weight less than $i-1$, and finds the minimum point for a fraction of more than $\frac{j}{i+1} + \frac{1}{\sqrt{i+1}}$ of the paths from 1^n to the third level. We fix a path p' from 1^n to a vertex v of weight $i+1$ such that $\mathcal{A}_2$ succeeds for a fraction of more than $\frac{j}{i+1} + \frac{1}{\sqrt{i+1}}$ of the functions in $\mathcal{P}'_{n,n-2}$ – the functions $p'p''$ where p'' is a path from v to a vertex of weight 3.

As before, $\mathcal{A}_2$ succeeds only for paths that are excluded. By Proposition 4.1 and the number of queries asked by $\mathcal{A}_2$, the fraction of paths in $\mathcal{P}'_{n,n-2}$ that are excluded is at most

$$j \cdot \frac{1}{i+1} + \frac{i\sqrt{i+1}}{4} \cdot \frac{2}{i(i+1)} + \frac{(i-1)i\sqrt{i+1}}{12} \cdot \frac{6}{(i-1)i(i+1)} \;=\; \frac{j}{i+1} + \frac{1}{\sqrt{i+1}}.$$

This contradicts the assumption regarding $\mathcal{A}_2$ and hence the lemma follows. ■

THEOREM A.1. *For any* randomized *algorithm $\mathcal{A}$ and any concept class $\mathcal{F}$, such that $\mathcal{P}_{n,n-2} \subseteq \mathcal{F}$,*

$$\mathrm{RMC}(\mathcal{A}, \mathcal{F}) \geq \frac{n^2}{4} - 8 \cdot n^{3/2}.$$

Proof. Assume that the expected number of queries that $\mathcal{A}$ asks is less than $n^2/4 - 8 \cdot n^{3/2}$. Let i be the index we fixed above. Using Lemma A.2 we give a lower bound on S_i:

$$\begin{aligned}
S_i \;&=\; \mathrm{E}(C_i) \;=\; \sum_{j=1}^{\infty} j \cdot \Pr[C_i = j] \\
&\geq\; (i+2) \cdot \Pr[C_i \geq i+2] + \sum_{j=1}^{i+1} j \cdot (\Pr[C_i \leq j] - \Pr[C_i \leq j-1]) \\
&\geq\; (i+1) \cdot \Pr[C_i \geq i+2] + (i+1)\Pr[C_i \leq i+1] - \sum_{j=0}^{i} \Pr[C_i \leq j] \\
&\geq\; i+1 - \sum_{j=0}^{i} \left(\frac{j}{i+1} + \frac{8}{\sqrt{i+1}} \right) \\
&>\; \frac{i+2}{2} - 8\sqrt{i+1}.
\end{aligned}$$

This contradicts the fact that i satisfies (A.1), and the theorem follows. ■

ACKNOWLEDGMENT

We would like to thank the anonymous referee for his contribution to the presentation of this paper and in particular for significantly simplifying the proof of Theorem 3.1. We would also like to thank the COLT '98 committee (and referees) for valuable comments.

REFERENCES

1. D. Angluin. Learning k-term DNF formulas using queries and counterexamples. Technical Report YALEU/DCS/RR-559, Department of Computer Science, Yale University, 1987.
2. D. Angluin. Learning regular sets from queries and counterexamples. *Information and Computation*, 75:87–106, 1987.
3. D. Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, 1988.
4. D. Angluin. Negative results for equivalence queries. *Machine Learning*, 5:121–150, 1990.
5. D. Angluin, M. Frazier, and L. Pitt. Learning conjunctions of Horn clauses. *Machine Learning*, 9:147–164, 1992.
6. D. Angluin, L. Hellerstein, and M. Karpinski. Learning read-once formulas with queries. *J. of the ACM*, 40:185–210, 1993.

7. D. Angluin, M. Kriķis, R. E. Sloan, and G. Turán. Malicious omissions and errors in answers to membership queries. *Machine Learning*, 28:211–255, 1997.
8. D. Angluin and D. K. Slonim. Randomly fallible teachers: learning monotone DNF with an incomplete membership oracle. *Machine Learning*, 14(1):7–26, 1994.
9. A. Beimel, F. Bergadano, N. H. Bshouty, E. Kushilevitz, and S. Varricchio. On the applications of multiplicity automata in learning. In *Proc. of the 37th Annu. IEEE Symp. on Foundations of Computer Science*, pages 349–358, 1996. Journal version: *J. of the ACM*, 47(3):506-530, 2000.
10. F. Bergadano, D. Catalano, and S. Varricchio. Learning sat-k-DNF formulas from membership queries. In *Proc. of the 28th Annu. ACM Symp. on the Theory of Computing*, pages 126–130, 1996.
11. A. Blum, P. Chalasani, S. A. Goldman, and D. K. Slonim. Learning with unreliable boundary queries. In *Proc. of 8th Annu. ACM Conf. on Comput. Learning Theory*, pages 98–107, 1995. Journal version: *J. of Computer and System Sciences*, 56(2):209–202, 1998.
12. A. Blum and S. Rudich. Fast learning of k-term DNF formulas with queries. In *Proc. of the 24th Annu. ACM Symp. on the Theory of Computing*, pages 382–389, 1992. Journal version: *J. of Computer and System Sciences*, 51(3):367–373, 1995.
13. N. H. Bshouty. Exact learning via the monotone theory. In *Proc. of the 34th Annu. IEEE Symp. on Foundations of Computer Science*, pages 302–311, 1993. Journal version: *Information and Computation*, 123(1):146–153, 1995.
14. N. H. Bshouty. Simple learning algorithms using divide and conquer. In *Proc. of 8th Annu. ACM Conf. on Comput. Learning Theory*, pages 447–453, 1995. Journal version: *Computational Complexity*, 6:174–194, 1997.
15. N. H. Bshouty and R. Cleve. On the exact learning of formulas in parallel. In *Proc. of the 33rd Annu. IEEE Symp. on Foundations of Computer Science*, pages 513–522, 1992.
16. N. H. Bshouty, S. A. Goldman, T. R. Hancock, and S. Matar. Asking questions to minimize errors. *J. of Computer and System Sciences*, 52(2):268–286, 1996.
17. M. Clausen, A. Dress, J. Grabmeier, and M. Karpinski. On zero-testing and interpolation of k-sparse multivariate polynomials over finite fields. *Theoretical Computer Science*, 84(2):151–164, 1991.
18. S. Goldman and H. Mathias. Learning k-term DNF formulas with an incomplete membership oracle. In *Proc. of 5th Annu. ACM Workshop on Comput. Learning Theory*, pages 77–85, 1992.
19. T. Hegedüs. Generalized teaching dimensions and the query complexity of learning. In *Proc. of 8th Annu. ACM Conf. on Comput. Learning Theory*, pages 108–117, 1995.
20. L. Hellerstein, K. Pillaipakkamnatt, V. Raghavan, and D. Wilkins. How many queries are needed to learn? *J. of the ACM*, 43(5):840–862, 1996.
21. E. Kushilevitz. A simple algorithm for learning $O(\log n)$-term DNF. In *Proc. of 9th Annu. ACM Conf. on Comput. Learning Theory*, pages 266–269, 1996. Journal version: *Inform. Process. Lett.*, 61(6):289–292, 1997.
22. W. Maass and G. Turán. On the complexity of learning from counterexamples. In *Proc. of the 30th Annu. IEEE Symp. on Foundations of Computer Science*, pages 262–273, 1989.
23. W. Maass and G. Turán. On the complexity of learning from counterexamples and membership queries. In *Proc. of the 31st Annu. IEEE Symp. on Foundations of Computer Science*, volume I, pages 203–210, 1990.
24. W. Maass and G. Turán. Lower bound methods and separation results for on-line learning models. *Machine Learning*, 9:107–145, 1992.
25. M. Naor, L. J. Schulman, and A. Srinivasan. Splitters and near-optimal derandomization. In *Proc. of the 36th Annu. IEEE Symp. on Foundations of Computer Science*, pages 182–191, 1995.
26. R. L. Rivest and R. E. Schapire. Inference of finite automata using homing sequences. *Information and Computation*, 103:299–347, 1993.
27. R. M. Roth and G. M. Benedek. Interpolation and approximation of sparse multivariate polynomials over GF2). *SIAM Journal on Computing*, 20(2):291–314, 1991.
28. R. E. Schapire and L. M. Sellie. Learning sparse multivariate polynomials over a field with queries and counterexamples. In *Proc. of 6th Annu. ACM Conf. on Comput. Learning Theory*, pages 17–26, 1993. Journal version: *J. of Computer and System Sciences*, 52(2):201–213, 1996.
29. A. C. Yao. Lower bounds by probabilistic arguments. In *Proc. of the 24th Annu. IEEE Symp. on Foundations of Computer Science*, pages 420–428, 1983.